UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR DE ECONOMIA E GESTÃO

Doutoramento em Matemática Aplicada à Economia e à Gestão

Uma variante do problema da floresta de Steiner em grafos com aplicações em biologia da conservação

Raúl Massano Brás

Orientador: Professor Doutor Jorge Orestes Lasbarrères Cerdeira
Co-orientadora: Professora Doutora Leonor Almeida Leite Santiago Pinto

Presidente do Júri:
Reitor da Universidade de Lisboa
Vogais:

- Doutor Jorge Orestes Lasbarrères Cerdeira, Professor Catedrático da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa;
- Doutor Miguel Fragoso Constantino, Professor Auxiliar da Faculdade de Ciências da Universidade de Lisboa;
- Doutora Graça Maria Marques da Silva Gonçalves, Professora Auxiliar da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa;
- Doutora Maria Cristina Saraiva Requejo Agra, Professora Auxiliar da Universidade de Aveiro;
- Doutora Leonor Almeida Leite Santiago Pinto, Professora Auxiliar do Instituto Superior de Economia e Gestão da Universidade de Lisboa.

Julho de 2013

Agradecimentos

Ao meu orientador, Professor Doutor Jorge Orestes Cerdeira, pelo seu empenho e dedicação no acompanhamento da elaboração da tese. Sem as suas contribuições, sempre oportunas e esclarecedoras, este trabalho não teria sido possível.

À co-orientadora, Professora Doutora Leonor Santiago Pinto, pela sua disponibilidade permanente para esclarecer dúvidas e rever o trabalho desenvolvido.

Aos docentes do programa de doutoramento em MAEG e em particular à coordenadora, Professora Doutora Margarida Vaz Pato, pelo apoio sempre demonstrado.

Aos co-autores dos artigos publicados durante a elaboração da tese: Jorge Orestes Cerdeira, Diogo Alagador, Maria Triviño, Mar Cabeza e Miguel Araújo. Agradeço em particular ao Doutor Diogo Alagador, pela disponibilização dos dados reais utilizados na tese e pela análise crítica das soluções obtidas com estes.

Ao CEMAPRE, Centro de Estudos de Matemática Aplicada à Economia e à Gestão, pela disponibilização dos meios informáticos necessários para obtenção dos resultados apresentados na tese.

Ao Instituto Superior de Economia e Gestão, pelas facilidades concedidas durante o período de elaboração da tese.

À minha família pelo apoio e incentivo permanentes.

Abstract

Habitat fragmentation is a serious threat for the sustainability of species. Thus, the identification of effective linkages to connect valuable ecological units is an important issue in conservation biology. The design of effective linkages should take into account that areas which are adequately permeable for some species' dispersal may act as obstructions for other species. The determination of minimum cost effective linkages is a generalization of both node-weighted Steiner tree and node-weighted Steiner forest problems. The thesis presents and compares formulations and heuristics to this problem. The heuristics were specially conceived to handle large instances that occur in conservation biology. The heuristics are compared using both simulated and real data for the Iberian Peninsula. Since the node weighted Steiner forest problem is a special case of the problem studied, the proposed heuristics are also compared to a well established heuristic for this case. The dissertation resulted in the development of an open source application that was made available to the scientific community.

Keywords: Combinatorial optimization, Graphs, Heuristics, Minimum Steiner Trees, Connectivity, Integer Programming.

Resumo

A fragmentação de habitats é uma ameaça séria à sustentabilidade das espécies ecológicas. Assim, a identificação de ligações eficientes entre unidades ecológicas, é uma questão importante em biologia da conservação. O estabelecimento de ligações eficientes, deve levar em conta que as áreas que são adequadamente permeáveis para a dispersão de algumas espécies, podem agir como obstáculos para outras. A determinação de ligações eficazes, com custo mínimo, é uma generalização dos problemas da árvore Steiner com custos nos nós e da floresta de Steiner com custos dos nós. A tese apresenta e compara formulações e heurísticas para este problema. As heurísticas foram especialmente concebidas para lidar com instâncias de grande dimensão, que ocorrem em biologia da conservação. As heurísticas são comparadas utilizando dados simulados e reais para a Península Ibérica. Uma vez que o problema da floresta de Steiner com custos nos nós é um caso particular do problema estudado, as heurísticas propostas são também comparadas com uma heurística bem conhecida para este caso. A elaboração da tese, levou ao desenvolvimento de uma aplicação informática de código aberto, que foi colocada à disposição da comunidade científica.

Palavras chave: Optimização Combinatória, Grafos, Heurísticas, Árvores Mínimas de Steiner, Conexidade, Programação Inteira.

Conteúdo

Lista de Figuras

Lista de Tabelas

Preâmbulo

O trabalho apresentado nesta dissertação tem origem na minha participação num estudo sobre o desenho de corredores entre áreas protegidas na Península Ibérica, por forma a garantir a conexidade destas. Este trabalho decorreu nos anos de 2009 e 2010 e deu origem a um artigo publicado na revista Landscape Ecology [Alagador et al., 2012].

A definição formal do problema foi apresentada na conferência *EURO XXIV* em 2010 (A modified Steiner forest problem with applications in conservation biology, R. Brás, J. O. Cerdeira, D. Alagador, M. Triviño, M. Cabeza, M. Araújo, Actas da conferência EURO XXIV, pág. 185).

Durante a redacção da dissertação, foi publicada nas actas da conferência *Advancement of Artificial Intelligence* de 2011 uma comunicação de Lai et al. [2011] que trata precisamente do problema que é o tema da presente dissertação. Tivemos, eu e os orientadores da dissertação, conhecimento desta publicação em meados de 2012, durante a fase final de redacção da dissertação. O problema tratado na comunicação citada e uma das heurísticas são muito semelhantes ao que apresentamos em 2010.

Lai et al. [2011] designaram o problema estudado nesta dissertação por "Steiner Multigraph Problem" e apresentaram uma formulação baseada em fluxos multi-produto e duas heurísticas.

A primeira heurística resolve um problema de Steiner em cada iteração, recorrendo ao algoritmo de Dreyfus-Wagner [Dreyfus and Wagner, 1971] e obtém a solução final fazendo a união das soluções obtidas em cada iteração. Esta heurística é semelhante à heurística *tipo a tipo* que apresentamos na conferência *EURO XXIV* em 2010, embora a dimensão das instâncias dos problemas que trabalhamos não fossem compatíveis com a utilização do algoritmo de Dreyfus-Wagner. No nosso caso, usou-se um método heurístico para obter a solução dos problemas de Steiner em cada uma das iterações.

A segunda é uma heurística primal-dual adaptada da heurística de De-

maine et al. [2009] para o problema da floresta de Steiner com custos nos nós, sendo igual à que é apresentada na dissertação. O facto de a mesma heurística ser proposta em trabalhos independentes não é de estranhar dado que resulta da adequação óbvia da heurística de Demaine et al. [2009] ao problema objecto desta dissertação.

Lai et al. [2011] apresentam resultados computacionais para problemas de pequena dimensão, com dados simulados e dados reais para a criação de corredores ecológicos para duas espécies animais no estado de Montana nos EUA. Para os dados reais a heurística primal-dual foi a mais rápida e com desvios para o óptimo relativamente pequenos.

Durante a elaboração da presente dissertação foram publicados dois artigos que incorporam o trabalho desenvolvido no seu âmbito: Alagador et al. [2012] e Brás et al. [2013]. Agradeço aos autores a contribuição decisiva que deram para os resultados apresentados na dissertação.

Capítulo 1

Introdução

Em biologia da conservação a fragmentação dos habitats é considerada como uma das principais causas do declínio da biodiversidade [Brooks et al., 2002, Hanski, 2005], com as consequências negativas que daí advêm para a vida no planeta. Para minimizar o problema da fragmentação convém estabelecer corredores entre áreas em que habitem espécies cuja conservação seja considerada importante (e.g., áreas protegidas).

A criação de corredores entre áreas seleccionadas tem como objectivo o estabelecimento de ligações capazes de sustentar os fluxos ecológicos (e.g., dispersão de espécies, migrações, ajustamentos espaciais a alterações climáticas) entre zonas de valor ecológico elevado [Merriam, 1984].

O desenho de corredores procura ligar, com custo mínimo, as áreas em que habitam as espécies relevantes. Recorrendo à teoria dos grafos, podemos considerar que cada célula de terreno é um nó do grafo e que as arestas representam ligações entre células adjacentes. Atribuindo a cada aresta um custo representando a dificuldade de movimentação no terreno e considerando o sub-conjunto de nós a interligar como terminais, o problema designa-se por árvore mínima de Steiner em grafos. Esta abordagem foi introduzida por Sessions [1992] no planeamento espacial da conservação de espécies.

Quando consideramos a ligação de unidades ecológicas de diferentes tipos (e.g. áreas que abrigam grupos de espécies com requisitos de mobilidade distintos) pode acontecer que células de terreno que são apropriadas para uma espécie, actuem como barreiras para outras. Nesta situação devemos desenhar os corredores por forma a minimizar o custo das ligações, garantindo ao mesmo tempo que as células de terreno seleccionadas não constituem barreiras à movimentação das espécies dos diferentes tipos considerados [Alagador

et al., 2012]. Classificar células de terreno como áreas protegidas tem custos económicos elevados pelo que se impõe desenvolver métodos para identificar ligações eficientes entre unidades ecológicas de tipos diferentes, garantindo que unidades do mesmo tipo são ligadas através de células adequadas para estas.

A solução para o problema descrito não pode ser obtida através de uma árvore mínima de Steiner, dado que esta não toma em consideração a possibilidade de cada um dos nós do grafo poder ser ou não utilizado nos caminhos que ligam os diferentes tipos de terminais. Voltando a considerar como nós terminais de um grafo as áreas (ou espécies) a ligar, a presença de tipos diferentes aponta para o problema da ligação de vários sub-conjuntos de terminais (tantos quantos os tipos). Se não houvesse restrições na utilização dos nós (alguns não são adequados para algumas espécies) o problema reduzia-se ao da floresta de Steiner em grafos com pesos nos nós. O exemplo da figura 1.1 mostra que estamos de facto perante uma variante deste problema.

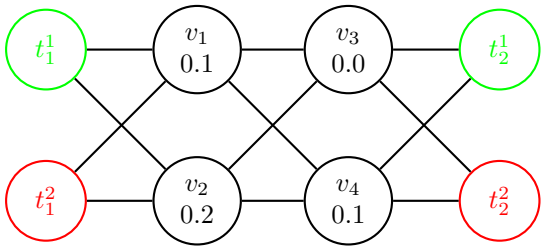
No exemplo temos dois tipos diferentes de unidades ecológicas, T^1 e T^2, com duas áreas de ocorrência cada $t_1^1,\ t_2^1$ e $t_1^2,\ t_2^2$ respectivamente. Existem ainda 4 nós, v_1 a v_4, para possíveis ligações. O nó v_1 não pode ser usado pelo tipo 1 e o nó v_3 não pode ser usado pelo tipo 2. Na figura 1.1b mostra-se, a azul, a solução para o problema da Floresta de Steiner (que admite que qualquer nó pode ser usado) com um custo de 0.1 e na figura 1.1c, a vermelho e verde, a solução óptima com um custo de 0.4.

O trabalho apresentado nesta dissertação teve origem na participação num estudo de um caso real. Tratou-se de estudar a interligação das áreas protegidas da Península Ibérica, tendo estas sido classificadas em quatro tipos diferentes de acordo com as suas características climáticas. A cada célula de 1×1 Km foram atribuídos índices de dissemelhança (um para cada tipo de reservas), não podendo estas ser usadas para ligar reservas do tipo k se o respectivo índice fosse maior que um valor previamente fixado. O grafo associado a este caso real tem 580.696 nós e 681 terminais, o que levou a que a ênfase do trabalho fosse colocada na obtenção de algoritmos capazes de resolver problemas de grande dimensão. Para mais detalhes sobre estes dados ver Alagador et al. [2012].

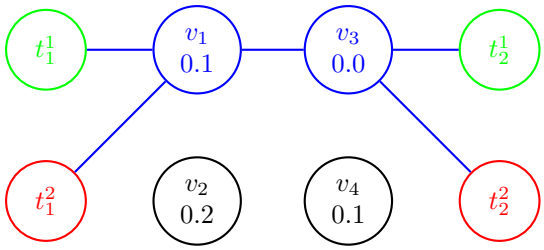
Para além de estudar e obter heurísticas capazes de resolver o problema, outro objectivo do trabalho foi o desenvolvimento de uma aplicação informática, disponibilizada para a comunidade científica, capaz de obter soluções a partir dos métodos heurísticos desenvolvidos.

O problema objecto de estudo será designado por Problema de Ligações

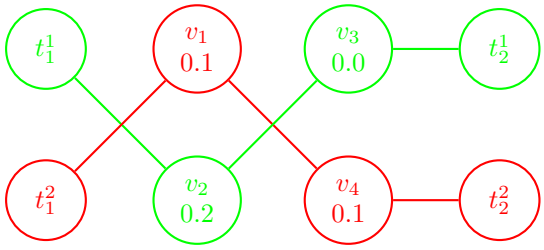
Figura 1.1: Exemplo do problema



(a) Grafo de exemplo



(b) Solução da floresta de Steiner



(c) Solução do problema com restrições na utilização dos nós

Multi Tipo (PLMT). Apresenta-se em seguida a sua definição formal.

Considere-se um grafo não orientado $G = (V, E)$, onde os nós representam células de terreno de uma dada região e as arestas representam adjacências entre pares de células. Dados τ tipos diferentes (e.g. áreas protegidas com características climáticas distintas ou espécies animais com habitats diferenciados), para $k = 1, 2, \ldots, \tau$ considerem-se subconjuntos $T^k \subset V$ com pelo menos dois nós cada, designados por *terminais* do tipo k, que se pretende ligar, e um subconjunto de nós $V^k \subset V$ que inclui os nós em T^k e os nós que podem ser usados para ligar os terminais em T^k. Para todo o subconjunto de nós $S \subseteq V$, seja $S^k = S \cap V^k$ e represente-se por $\langle S^k \rangle$ o sub-grafo de G induzido por S^k.

S é *uma solução admissível* se todos os terminais de T^k pertencerem à mesma componente conexa de $\langle S^k \rangle$, para todo o $k = 1, 2, \ldots, \tau$. Considere-se um peso não negativo w_v associado a cada nó (e.g. custo de classificar uma célula de terreno como protegida). PLMT procura uma solução admissível de custo mínimo (i.e., uma solução que minimize a soma dos pesos dos nós).

Se todos os terminais forem do mesmo tipo (i.e., $\tau = 1$), PLMT reduz-se ao problema de Steiner em grafos com custos nos nós [Klein and Ravi, 1995]. Se todos os conjuntos de nós não terminais forem iguais, i.e. $V^k = V$, e os τ conjuntos de terminais forem diferentes, o problema reduz-se ao da floresta de Steiner com pesos nos nós [Duin and Volgenant, 1987].

Existem diversas variantes e extensões dos problemas de Steiner com pesos nos nós, associadas a aplicações práticas, e uma considerável literatura dedicada a estes problemas. Exemplos de aplicações incluem "multicast routing" onde os custos dos "routers" são tomados em consideração, [Angelopoulos, 2006], recuperação eficiente de falhas em redes de energia eléctrica [Guha et al., 1999], e análise da interacção de proteínas [Betzler, 2006].

PLMT é uma generalização do problema de Steiner em grafos com pesos nos nós e do problema da floresta de Steiner com pesos nos nós. O objectivo da dissertação é encontrar e testar métodos que permitam resolver instâncias de dimensão moderada a elevada do problema, tais como as que ocorrem no desenho de corredores para minimizar a fragmentação no contexto da biologia da conservação.

O capítulo 2 contém uma revisão bibliográfica de problemas relacionados com o objecto da dissertação. O capítulo 3 introduz e compara duas formulações baseadas em fluxos multi-produto e uma em cortes de cobertura para o PLMT. Serão descritas e analisadas três heurísticas para obter soluções para o problema no capítulo 4. No capítulo 5 apresentam-se resultados dos testes computacionais efectuados para aferir os tempos de execução e qualidade dos resultados das três heurísticas. O capítulo 6 descreve a aplicação informática desenvolvida para implementar os resultados obtidos no decurso da elaboração da dissertação. Finalmente no capítulo 7 apresentam-se algumas conclusões.

Capítulo 2

Problemas relacionados

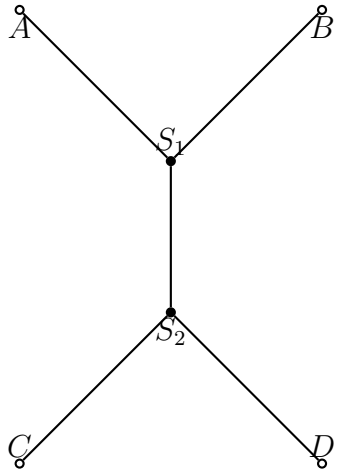
2.1 Nota histórica

O problema da árvore de Steiner deve o seu nome ao matemático suíço Jacob Steiner (1796-1863), embora tenha sido proposto por Fermat e posteriormente definido por Gauss, numa carta dirigida ao seu aluno Schumacher [Vazirani, 2003]. A designação "problema de Steiner" foi popularizada por Courant and Robbins [1979]. O problema original consiste no seguinte: interligar um conjunto dado de pontos no plano, directamente ou através de pontos introduzidos, de forma a que o comprimento total das ligações seja mínimo. Este problema designa-se por árvore de Steiner euclidiana. Os pontos dados designam-se por terminais e os introduzidos por pontos de Steiner. A Figura 2.1 mostra uma solução para este problema com 4 terminais: A, B, C e D. Os pontos S_1 e S_2 são os pontos de Steiner.

Ao longo dos anos foram introduzidas diversas variantes do problema. A árvore de Steiner rectilínear é um problema em que a distância euclidiana é substituída pela distância rectilínear e tem aplicações no desenho de circuitos integrados.

Este trabalho considera apenas variantes do problema de Steiner em grafos pelo que analisaremos, com alguma profundidade, este caso.

Figura 2.1: Problema de Steiner com 4 pontos



2.2 Problema de Steiner em grafos

No caso em que os terminais e os pontos de Steiner são nós de um grafo temos o problema de Steiner em grafos.

2.2.1 Custos nas arestas

O problema de Steiner em Grafos foi introduzido por Hakimi [1971]. Dado um grafo não orientado $G = (V, E)$, uma função $c : E \to Q^+$ de custos não negativos nas arestas e um subconjunto $T \subseteq V$, determinar a árvore de custo mínimo que contém T. Os nós em T são os terminais, enquanto que os restantes da solução são os pontos de Steiner. Se $T = V$ o problema é o da árvore de suporte mínima e se $|T| = 2$ o problema é o do caminho mais curto. Este problema é NP-completo [Karp, 1972], excepto para os dois casos particulares referidos.

Algoritmos exactos

No artigo citado de Hakimi é apresentado um algoritmo para resolver o problema. Este recorre à enumeração das árvores de suporte mínimo de subgrafos de G induzidos por subconjuntos $W \subseteq V$ tais que $T \subseteq W$. Este algoritmo foi depois melhorado por Lawler [1976]. A complexidade do algoritmo de Lawler é $0\,(p^2 2^{n-p} + n^3)$, onde $p = |T|$ e $n = |V|$.

Um dos algoritmos exactos mais conhecidos é o algoritmo de programação dinâmica de Dreyfus and Wagner [1971]. O algoritmo calcula recursivamente árvores óptimas $A(X \cup \{v\})$ para todo o $X \subseteq T$ e $v \in V \setminus X$. v liga a algum w de $A(X \cup \{v\})$ através do caminho mais curto P_{vw}. Tomando uma partição não trivial $X = X^1 \cup X^2$ pode escrever-se $A(X \cup \{w\}) = A(X^1 \cup \{w\}) \cup A(X^2 \cup \{w\})$. Como $A(X \cup \{v\}) = P_{vw} \cup A(X \cup \{w\})$ pode escrever-se, abusando um pouco da notação,

$$A(X \cup \{v\}) = \min\ P_{vw} \cup A\left(X^1 \cup \{w\}\right) \cup A\left(X^2 \cup \{w\}\right) \tag{2.1}$$

A equação 2.1 permite-nos calcular recursivamente todas as árvores óptimas $A(X \cup \{v\})$ para $v \in V$ e $X \subseteq T$. A complexidade do algoritmo de Dreyfus-Wagner é $O(n3^p + n^2 2^p + n^3)$. Em 2007 Fuchs et al. [2007], a quem se deve a descrição do algoritmo acima, introduziram uma alteração ao algoritmo melhorando o seu desempenho para $O^*(2.684^p)$ (a notação O^* significa que foram suprimidos termos de ordem polinomial).

Formulações de Programação Inteira

O problema pode ser formulado utilizando Programação Inteira. Aneja [1980] utilizou uma formulação de conjunto de cobertura para este efeito.

Seja $G = (V, E)$ um grafo não orientado com pesos nas arestas $c_j > 0$ e considere-se uma partição de V, $X, \overline{X}$, tal que $T \cap X \neq \emptyset$ e $T \cap \overline{X} \neq \emptyset$.

Designe-se por P o conjunto das arestas que constituem o corte entre X e $\overline{X}$ e sejam $P_1, P_2, \cdots, P_\rho$ os conjuntos de cortes obtidos considerando todas estas partições.

Defina-se:

$$a_{ij} = \begin{cases} 1 & \text{se a aresta } j \in P_i \\ 0 & \text{caso contrário} \end{cases}$$

para $i = 1, 2, \cdots, \rho$ e $j = 1, 2, \cdots, m$, onde m é o número de arestas.

Aneja formulou o problema de Steiner através de:

$$\min \quad \sum_{j=1}^{m} c_j x_j \tag{2.2a}$$

sujeito a

$$\sum_{j=1}^{m} a_{ij} x_j \geq 1 \quad \text{para } i = 1, 2, \cdots, \rho \tag{2.2b}$$

$$x_j \in \{0, 1\} \quad \text{para } j = 1, 2, \cdots, m \tag{2.2c}$$

Seja x^* uma solução óptima do problema e $S^* = \{j \in E \mid x_j^* = 1\}$. S^* é solução óptima do problema de Steiner em grafos. Aneja começa por notar que S^* não contém ciclos dado que $c_j > 0$ para todas as arestas do grafo. Em segundo lugar, S^* contém apenas uma componente conexa que liga todos os terminais. Se assim não fosse a restrição correspondente a algum conjunto de corte P_i seria violada. Finalmente como S^* é obtido a partir da solução óptima, é uma árvore de custo mínimo que liga todos os terminais.

Embora o problema tenha um número exponencial de restrições, dependente de m, $|V|$ e $|T|$, Aneja mostra como obter a solução da relaxação linear de (2.2a)-(2.2c) sem enumerar explicitamente as restrições, recorrendo à técnica de geração de linhas. Resolve a relaxação linear do problema para um sub-conjunto de linhas de (2.2b) e se a solução não for óptima introduz uma nova linha e repete o processo. Tirando partido das propriedades do problema de cobertura obteve um algoritmo que permite obter limites inferiores e superiores para a solução do problema e, eventualmente, o óptimo.

Em 1984 Wong [1984] apresentou uma formulação de programação inteira baseada em fluxos multi-produto. Dado que o problema em grafos não orientados pode ser transformado num problema em grafos orientados, substituindo cada aresta por dois arcos, com direcções opostas e o mesmo custo, Wong apresentou a sua formulação para o segundo caso. A solução do problema orientado é uma arborescência e esta corresponde a uma árvore que é solução do problema original.

Considere-se um grafo orientado $G = (V, A)$ com $V = \{1\} \cup T \cup S$ onde $\{1\}$ designa um terminal arbitrário, T os restantes terminais e S os possíveis nós de Steiner, i.e. nós utilizáveis para interligar os terminais. A cada arco (i, j) de A está associado um custo não negativo c_{ij}.

A formulação de Wong é:

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{2.3a}$$

sujeito a:

$$\sum_{h \in V} f_{ih}^k - \sum_{j \in V} f_{ji}^k = \begin{cases} 1 & \text{se } i = 1 \\ -1 & \text{se } i = k \\ 0 & \text{se } i \neq 1, k \end{cases} \qquad k \in T,\, i \in V \tag{2.3b}$$

$$f_{ij}^k \leq x_{ij}, \quad (i,j) \in A, \;\; k \in T \tag{2.3c}$$

$$f_{ij}^k \geq 0, \quad (i,j) \in A, \;\; k \in T \tag{2.3d}$$

$$x_{ij} \in \{0,1\}, \quad (i,j) \in A \tag{2.3e}$$

onde x_{ij} é uma variável binária que indica se o arco (i,j) é usado na solução e f_{ij}^k é a quantidade de fluxo entre os nós 1 e $k \in T$ no arco (i,j) (quantidade de produto k). As restrições (2.3c) podem ser escritas de forma condensada:

$$\sum_{k \in T \setminus \{1\}} f_{ij}^k \leq n x_{ij} \quad (i,j) \in A$$

com $n = |T|$. A opção por (2.3c) deve-se segundo Wong ao facto de, embora tendo um maior número de restrições, permitir obter algoritmos mais eficientes.

Nesta formulação, a restrição (2.3b) garante que sai uma unidade de fluxo do nó escolhido como origem com destino a cada um dos restantes terminais e (2.3c) que o fluxo só atravessa arcos que estejam na solução.

Para comparar a sua formulação com a de Aneja, Wong reescreveu (2.2a)-(2.2c) para o caso de um grafo orientado obtendo:

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{2.4a}$$

sujeito a:

$$\sum_{i \in H} \sum_{j \in \overline{H}} x_{ij} \geq 1 \quad \forall H \subseteq V : 1 \in H,\; \bar{H} = V \setminus H,\; \overline{H} \cap T \neq \emptyset \tag{2.4b}$$

$$x_{ij} \in \{0, 1\} \tag{2.4c}$$

Mostrando que as soluções das relaxações lineares de (2.3a)-(2.3e) e de (2.4a)-(2.4c) têm o mesmo valor, Wong desenvolveu um algoritmo diferente do de Aneja para obter uma solução aproximada. Enquanto o primeiro trabalha sobre uma relaxação linear do problema, Wong trabalha sobre o dual da relaxação linear da sua formulação.

O dual da relaxação linear de (2.3a)-(2.3e) é:

$$\max \sum_{k \in T} \left(v_k^k - v_1^k\right) \tag{2.5a}$$

sujeito a:

$$v_j^k - v_i^k - w_{ij}^k \leq 0, \quad k \in T, \;\; i, j \in V \tag{2.5b}$$

$$\sum_{k \in T} w_{ij}^k \leq c_{ij} \quad (i, j) \in A \tag{2.5c}$$

$$w_{ij}^k \geq 0 \tag{2.5d}$$

Wong apresentou um algoritmo dual-ascendente (um tipo particular de algoritmo primal-dual [Goemans and Williamson, 1997]). A cada arco a está associado um custo reduzido $\overline{c}_a$, que tem inicialmente o valor c_a. Arcos com custo reduzido igual a zero designam-se por saturados e induzem um grafo de saturação $G_S = (V, A_S)$, onde A_S representa o conjunto de arcos saturados. Inicialmente A_S é vazio dado que por hipótese $c_a > 0$ para todo o $a \in A$. Seja C uma componente fortemente conexa de G_S. C é uma componente raiz se verifica:

1. C contém um terminal;
2. C não contém a origem escolhida ($\{1\}$ na notação usada por Wong);
3. Não existe nenhum terminal $t \notin C$ que esteja ligado a um elemento de C por um caminho em G_S.

Dada uma componente raiz C, sejam $W(C) \supseteq C$ o conjunto de nós ligados a C por caminhos em G_S e $\delta^-(W)$ o conjunto de arcos incidentes a W.

Como inicialmente $A_S = \emptyset$ cada terminal, excluindo a origem, corresponde a uma componente raiz. O algoritmo Dual-ascendente de Wong é:

1. $LB \leftarrow 0$
2. $\overline{c}_a \leftarrow c_a, \ \ \forall a \in A$
3. Enquanto existirem componentes raiz em G_S
   (a) Escolher uma componente raiz C
   (b) $W \leftarrow W(C)$
   (c) $\Delta \leftarrow \min_{a \in \delta^-(W)} \overline{c}_a$
   (d) $a^* \leftarrow \operatorname{argmin}_{a : a \in \delta^-(W)} \overline{c}_a$
   (e) $\overline{c}_a \leftarrow c_a - \Delta, \ \ \forall a \in \delta^-(W)$
   (f) $LB \leftarrow LB + \Delta$
   (g) $A_S \leftarrow A_S \cup a^*$
4. Devolver LB e A_S

onde LB é um limite inferior para o valor do óptimo.

Em cada iteração pelo menos um arco é saturado, reduzindo-se assim o número de componentes raiz até que o algoritmo termina.

Quando o algoritmo termina, G_S contém pelo menos um caminho orientado entre a origem e todos os outros terminais.

Para obter a solução do problema de Steiner orientado (e consequentemente do problema não orientado) Wong propõe o procedimento seguinte:

1. Seja Q o conjunto de nós de G_S para os quais existe um caminho a partir da origem. Calcular uma arborescência de suporte mínima no sub-grafo de G_S induzido por Q.
2. Eliminar as folhas que não sejam terminais.

Obtém-se assim uma solução do problema cujo valor é um limite superior para a solução óptima.

Esta formulação, e respectivo algoritmo, foi retomada recentemente por Drummond et al. [2009] que a adaptaram para computação distribuída com o objectivo de resolverem problemas relacionados com a emissão de vídeo em redes informáticas.

Magnanti and Raghavan [2005] estudaram problemas genéricos de conexidade apresentando formulações que incluem o problema da árvore de Steiner com pesos nas arestas. Partindo da formulação clássica por cortes:

$$\min \sum_{\{i,j\}\in E} c_{ij}x_{ij} \tag{2.6a}$$

sujeito a:

$$\sum_{\{i,j\}\in\delta(S)} x_{ij} \geq 1, \quad \begin{array}{l} \forall S \neq \emptyset \text{ e } S \subset V, \\ S \cap T \neq \emptyset, \\ (V \setminus S) \cap T \neq \emptyset \end{array} \tag{2.6b}$$

$$x_{ij} \in \{0,1\} \quad \forall \{i,j\} \in E \tag{2.6c}$$

e da formulação com fluxos:

$$\min \sum_{\{i,j\}\in E} c_{ij}x_{ij} \tag{2.7a}$$

sujeito a:

$$\sum_{j\in V} f^k_{ji} - \sum_{l\in V} f^k_{il} = \begin{cases} -1 & \text{se } i = t_1, \\ 1 & \text{se } i = k \\ 0 & \text{outros} \end{cases}, \quad \begin{array}{l} \forall i \in V, \\ k \in T \setminus \{t_1\} \end{array} \tag{2.7b}$$

$$\left.\begin{array}{l} f^k_{ij} \\ f^k_{ji} \end{array}\right\} \leq x_{i,j} \quad \forall \{i,j\} \in E \text{ e } k \in T \setminus \{t_1\} \tag{2.7c}$$

$$f_{ij}^k,\ f_{ji}^k \geq 0\ \ \forall \{i,j\} \in E \text{ e } k \in T \setminus \{t_1\} \tag{2.7d}$$

$$x_{ij} \in \{0,1\}\ \ \forall \{i,j\} \in E \tag{2.7e}$$

mostraram que as duas formulações são equivalentes.

Designem-se por $P_{cut}^{(2.6)}$ e $P_{flo}^{(2.7)}$ os poliedros definidos pela relaxação linear das restrições de (2.6) e (2.7) respectivamente.

Recorrendo ao teorema do fluxo-máximo, corte-mínimo, Magnanti and Raghavan [2005] provaram que as formulações (2.6) e (2.7) são equivalentes.

A formulação (2.6) pode ser melhorada recorrendo a uma formulação orientada, isto é, uma formulação que tem por base o mesmo grafo mas com as arestas substituídas por arcos. Para o caso da árvore de Steiner tem-se:

$$\min \sum_{\{i,j\} \in E} c_{ij} x_{ij} \tag{2.8a}$$

sujeito a:

$$\sum_{(i,j) \in \delta^-(S)} y_{ij} \geq 1, \quad \begin{array}{l} \forall S \neq \emptyset \text{ e } S \subset V, \\ t_1 \notin S \end{array} \tag{2.8b}$$

$$y_{ij} + y_{ji} \leq x_{ij}\ \ \forall \{i,j\} \in E \tag{2.8c}$$

$$y_{ij},\ y_{ji} \geq 0\ \ \forall \{i,j\} \in E \tag{2.8d}$$

$$x_{ij} \in \{0,1\} \tag{2.8e}$$

Mais uma vez o teorema do fluxo-máximo, corte-mínimo, permite obter uma formulação melhorada, equivalente a (2.8)

$$\min \sum_{\{i,j\}\in E} c_{ij}x_{ij} \tag{2.9a}$$

sujeito a:

$$\sum_{j\in V} f_{ji}^k - \sum_{l\in V} f_{il}^k = \begin{cases} -1 & \text{se } i = t_1, \\ 1 & \text{se } i = k \\ 0 & \text{outros} \end{cases}, \quad \begin{array}{l} \forall i \in V, \\ k \in T \setminus \{t_1\} \end{array} \tag{2.9b}$$

$$f_{ij}^k + f_{ji}^h \leq x_{ij} \;\; \forall \{i,j\} \in E \text{ e } h,k \in T \setminus \{t_1\} \tag{2.9c}$$

$$f_{ij}^k,\; f_{ji}^k \geq 0 \;\; \forall \{i,j\} \in E \text{ e } k \in T \setminus \{t_1\} \tag{2.9d}$$

$$x_{ij} \in \{0,1\} \;\; \forall \{i,j\} \in E \tag{2.9e}$$

.

Heurísticas

Como não existem algoritmos polinomiais para o problema e a dimensão dos casos práticos é por vezes muito grande, recorre-se a heurísticas para encontrar soluções.

Uma heurística óbvia é calcular a árvore de suporte mínima de $G = (V, E)$ e podar todas as folhas não terminais da árvore até que todas elas sejam nós terminais. Um exemplo devido a Rayward-Smith and Clare [1986] mostra que este procedimento ingénuo produz resultados arbitrariamente maus. Considere-se um grafo com

$$V = \{1, 2, \cdots, n\}$$

e

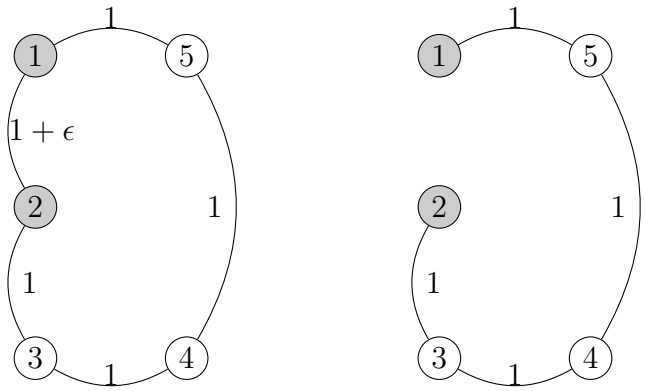
$$E = \{[i, j] : i = 1, \cdots, n-1,\ j = i+1\}$$

e definam-se os custos nas arestas através de:

$$c_{ij} = \begin{cases} 1+\epsilon & \text{se } i = 1, j = 2 \\ 1 & \text{outros} \end{cases}$$

considerando o conjunto de terminais $T = \{1, 2\}$ o custo da árvore obtida com esta heurística é $n-1$ enquanto que o custo da árvore de Steiner é $1+\epsilon$ (ver Figura 2.2).

Figura 2.2: Exemplo de aplicação da heurística



A qualidade de uma heurística pode medir-se pelo seu rácio de desempenho no pior caso (RD), medido por $c(H)/c(OPT)$ onde $c(H)$ designa o custo da solução dada pela heurística e $c(OPT)$ o custo da solução óptima. Outra medida importante é a sua complexidade algorítmica que nos dá uma ideia sobre o tempo de execução do algoritmo em função da dimensão do problema.

As primeiras heurísticas para o problema de Steiner, com análise do RD, datam do início dos anos 80 e dado o seu interesse histórico e simplicidade descrevem-se em seguida.

A primeira deve-se a Takahashi and Matsuyama [1980]. Trata-se de uma heurística construtiva que acrescenta sucessivamente nós terminais até obter uma solução admissível.

1. Iniciar com uma sub-árvore S_1 de G com um nó arbitrário $v \in T$. $k = 1$
2. Determinar o nó $v \in T \setminus S_k$ com o caminho mais curto para os nós em S_k e juntar o caminho a S_k. $k = k + 1$

3. Se $k < p$ voltar a 2, senão terminar.

A complexidade da heurística é de $O(pn^2)$ e o RD é de $2 - \frac{2}{p}$, cujo supremo é 2. n e p designam o número de nós e de terminais respectivamente.

No ano seguinte Kou et al. [1981] publicaram uma heurística baseada no cálculo da árvore de suporte mínima.

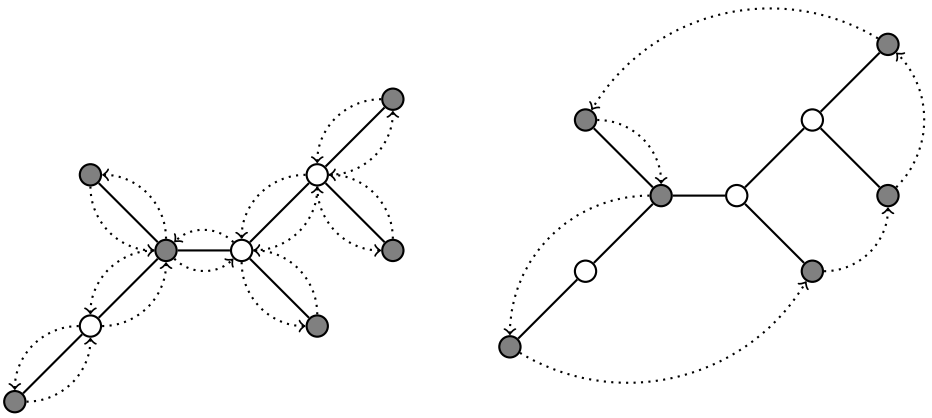
1. A partir do grafo $G(V, E)$ e do conjunto de terminais T, construir o grafo completo $G_1(T, E_1)$ onde os pesos de E_1 são dados pelo peso do caminho mais curto em G.
2. Calcular a árvore de suporte mínima de G_1
3. Construir o sub-grafo G_T de G substituindo as arestas da árvore de suporte mínima pelo correspondente caminho mais curto em G.
4. Calcular a árvore de suporte mínima de G_T.
5. Eliminar os nós não terminais de grau 1.

A complexidade e o RD são iguais aos da heurística anterior. No entanto em alguns casos esta heurística produz melhores resultados.

Para provar que esta heurística tem um factor de aproximação ao óptimo de 2, note-se que a redução do problema de Steiner em grafos ao do problema métrico de Steiner preserva o factor de aproximação [Vazirani, 2003]. Designa-se problema métrico o que é obtido através do fecho métrico do grafo, i.e. ao grafo completo obtido a partir de $G = (V, E)$ em que o custo de cada aresta $[u, v]$ é o do caminho mais curto entre u e v. O custo das arestas respeita a desigualdade triangular. Designe-se por $G_M(V, E_M)$ o fecho métrico de G e considere-se uma solução do problema métrico de Steiner de custo OPT. Duplicando as arestas desta árvore, obtém-se um caminho Euleriano (um caminho que usa cada aresta uma e uma só vez) de custo $2OPT$. Construa-se em seguida um caminho Hamiltoniano (um caminho que usa cada nó uma e uma só vez) curto-circuitando os nós não terminais e os terminais já utilizados (ver Figura 2.3). A desigualdade triangular implica que o custo deste caminho não é superior ao do caminho Euleriano. Retirando uma das arestas ao caminho obtém-se uma árvore de suporte mínimo dos terminais de custo não superior a $2OPT$.

Em 1983 Rayward-Smith [Rayward-Smith, 1983, Rayward-Smith and Clare, 1986] propôs um tipo de heurística diferente. Notando que se conhecermos

Figura 2.3: Caminho Hamiltoniano



os pontos de Steiner da solução, o problema se reduz ao cálculo da árvore de suporte mínima do grafo induzido por $T \cup S$, onde T designa os terminais e S os pontos de Steiner, propôs uma heurística para obter os pontos de S. O algoritmo começa com um conjunto de árvores $\mathcal{A}$, contendo inicialmente um terminal cada uma delas. Em cada iteração escolhe-se um nó usando uma função heurística $f(v)$ definida em função da distância entre v e os vários sub-grafos $\mathcal{A}$. O nó v obtido é usado para ligar duas das árvores em $\mathcal{A}$, para as quais $f(v)$ é mínimo, ligando-as a v através do caminho mais curto. A função $f(v)$ é definida por:

$$f(v) = \min_{1 \leq r \leq k-1} \left\{ \sum_{i=0}^{r} \frac{d\,(v, A_i)}{r} \right\}$$

onde $k = |\mathcal{A}|$ é o número de árvores, $A_0, \cdots, A_{k-1}$ são os conjuntos de nós das várias árvores em $\mathcal{A}$ e $d(v, A_i)$ é o custo mínimo dos caminhos mais curtos entre v e os nós de A_i.

A função $f(v)$ é uma medida da proximidade média do nó v às árvores num subconjunto de $\mathcal{A}$ contendo pelo menos dois elementos.

Ordenando os conjuntos A_i por ordem crescente de distância a v tem-se $d(v, A_0) \leq d(v, A_1) \leq \cdots \leq d(v, A_{k-1})$. Podemos agora escrever:

$$f_1(v) = \frac{d(v, A_0) + d(v, A_1)}{1}$$

$$f_2(v) = \frac{d(v,A_0) + d(v,A_1) + d(v,A_2)}{2} = \frac{f_1(v) + d(v,A_2)}{2}$$

$$f_3(v) = \frac{d(v,A_0) + d(v,A_1) + d(v,A_2) + d(v,A_3)}{3} = \frac{2f_2(v) + d(v,A_3)}{3}$$

$$\vdots$$

$$f_r(v) = \frac{(r-1)f_{r-1}(v) + d(v,A_r)}{r} = f_{r-1}(v) + \frac{d(v,A_r) - f_{r-1}(v)}{r} \quad (2.10)$$

claramente $f(v) = \min\{f_r(v) : 1 \leq r \leq k-1\}$. Para obter $f(v)$ não é necessário calcular recursivamente todos os $f_r(v)$. A equação (2.10) mostra que $f_i(v) < f_{i-1}(v)$ se e só se $d(v,A_i) < f_{i-1}(v)$. Por outro lado $d(v,A_i) \geq f_{i-1}(v) \Rightarrow f_{i+1}(v) \geq f_i(v)$ dado que:

$f_{i+1}(v) = f_i(v) + \frac{d(v,A_{i+1}) - f_i(v)}{i+1}$
e

$$\begin{aligned} d(v,A_{i+1}) - f_i(v) &= d(v,A_{i+1}) - f_{i-1}(v) - \tfrac{d(v,A_i) - f_{i-1}(v)}{i} \\ &\geq d(v,A_i) - f_{i-1}(v) - \tfrac{d(v,A_i) - f_{i-1}(v)}{i} \\ &\geq [d(v,A_i) - f_{i-1}(v)]\left(1 - \tfrac{1}{i}\right) \\ &\geq 0 \end{aligned}$$

Por indução podemos provar que $f_r(v)$ é não decrescente a partir do momento em que atinge o mínimo, bastando calcular $f_i(v)$ enquanto $d(v,A_i) < f_{i-1}(v)$, sendo o último valor o mínimo. O algoritmo de Rayward-Smith é:

1. $\mathcal{A} \leftarrow \{\{t\}\} \quad \forall t \in T$
2. Calcular $d(v,v') \quad \forall v, v' \in V$ e guardar os caminhos $R(v,v')$
3. Calcular $d(v,A)$ e $R(v,A) \quad \forall v \in V,\ A \in \mathcal{A}$
4. Calcular $f(v) \quad \forall v \in V.\ \alpha \leftarrow \arg(\min f(v))$
5. A α corresponde uma ordem dos elementos de $\mathcal{A}$: $A_0, A_1, \cdots, A_{k-1}$

6. $A_0 \leftarrow A_0 \cup A_1 \cup R(\alpha, A_0) \cup R(\alpha, A_1)$. Eliminar A_1
7. Se $|\mathcal{A}| \neq 1$ calcular $d(v, A_0)$ e $R(v, A_0)$ e voltar a 4.

Nos testes efectuados por Rayward-Smith este algoritmo mostrou melhorias significativas face aos anteriormente descritos.

As estes primeiros trabalhos seguiu-se uma longa série de artigos sobre heurísticas para este problema. Limitamos-nos aqui a destacar as que contribuíram para melhorias no RD.

Durante a década de 80 a razão de aproximação ao óptimo, no pior caso, manteve-se em 2. O algoritmo de Takahashi e Matsuyama tinha por base a ideia de fazer crescer uma árvore, juntando em cada iteração o caminho de menor custo para um nó de T que ainda não estivesse na solução. Em 1993 Zelikovsky [1993] generalizou esta ideia considerando os caminhos mais curtos para sub-árvores óptimas de subconjuntos de $k = 3$ elementos de T. Obteve assim um algoritmo com um RD de $11/6 \approx 1,84$ e complexidade $O(mn + p^4)$, onde $m = |E|$.

Em 1994 Berman and Ramaiyer [1994] colocaram a questão de saber se o algoritmo anterior podia ser melhorado considerando sub-árvores óptimas de $k > 3$ elementos. Tendo concluído que a resposta é negativa usaram uma técnica diferente de utilização de sub-árvores óptimas de dimensão $k \geq 3$. Para $k = 3$ obtiveram o mesmo RD que Zelikovsky [1993] mas com melhor desempenho em termos de velocidade de execução e para $k = 4$ melhoraram o RD anterior, atingindo um factor de aproximação de $16/9 \approx 1,78$.

Em 1996 Zelikovsky voltou ao tema, definindo uma metodologia a que chamou "Greedy contraction framework" [Zelikovsky, 1996]. A metodologia inclui como casos particulares, entre outros, as heurísticas de Takahashi e Matsuyama, a sua heurística de 1993 aqui chamada de "generalized greedy heuristic" e uma nova heurística a que chamou "relative greedy heuristic" (RGH). A nova heurística tem um factor de aproximação ao óptimo de $1 + \ln 2 \approx 1,693$. Não são apresentados detalhes computacionais.

Em 1997 Prömel e Steger publicaram um algoritmo aleatório, recorrendo a hipergrafos, com um factor de aproximação ao óptimo de 1,667 [Prömel and Steger, 1997].

No mesmo ano Karpinsky and Zelikovsky [1997] publicaram um novo artigo em que melhoraram os factores de aproximação dos algoritmos Berman-Ramayer (BR) e RGH. Notaram que em cada iteração ao maximizar o ganho total (na diminuição do custo da árvore de suporte mínima) se introduz uma

k-árvore (árvore de Steiner com, no máximo, k terminais sendo todos eles folhas da árvore) com todos os seus nós de $V \setminus T$ o que não conduz necessariamente à melhor solução na iteração corrente. Para evitar este problema introduziram um passo de pré-processamento que torna mais eficaz o algoritmo. Conseguiram reduzir o factor de aproximação ao óptimo do BR para 1,757 e do RGH para 1,644

Em 1999 Hougardy e Prömel publicaram um artigo com uma ideia nova [Hougardy and Prömel, 1999]. A ideia geral é aplicar iterativamente uma série de algoritmos ao resultado do seu antecessor. Em cada iteração a solução é a união da anterior com os pontos de Steiner obtidos. Findo o processo iterativo calcula-se uma árvore de suporte mínima da última solução. O artigo conclui que aplicando o processo iterativo ao algoritmo RGH se obtém um factor de aproximação ao óptimo de 1,598 depois de 11 iterações. O limite quando o número de iterações tende para infinito é de 1,588.

Este resultado só foi melhorado em 2005. Robins and Zelikovsky [2005] notaram que todos os algoritmos, com excepção do BR, escolhem componentes conexas (sub-árvores óptimas contendo k terminais) e depois contraemnas para formar a solução. Esta técnica não permite descartar uma componente já escolhida ainda que, mais tarde, se encontre uma melhor que entre em conflito com uma previamente aceite (duas componentes estão em conflito se houver pelo menos dois terminais comuns). A solução que encontraram foi contrair o menor número possível de componentes, por forma a que uma componente escolhida participe na solução, desde que não leve à exclusão de um número significativo de outras componentes. Com estes princípios criaram um algoritmo com um RD de $1 + \frac{\ln 3}{2} \approx 1.55$ e uma complexidade de $O\left(p^k(n-p)^{k-2} + kp^{2k+1}\log p\right)$. Neste artigo Robins e Zelikovsky mostraram ainda que para o caso de grafos quase-bipartidos (grafos em que não existem nós não terminais adjacentes) e grafos completos com custos 1 e 2 a heurística tem um RD de 1.28.

O melhor resultado até ao momento deve-se a Byrka et al. [2010]. Este algoritmo aplica uma técnica diferente de todos os anteriormente descritos. Utiliza uma relaxação linear de uma formulação por cortes do problema de Steiner e uma nova técnica designada por arredondamentos iterativos aleatórios para obter uma solução com um RD de 1,39.

2.2.2 Custos nos nós

Dado um grafo não orientado $G = (V, E)$, uma função $c : V \to Q_0^+$ de custos não negativos nos nós e um subconjunto $T \subseteq V$ de terminais, determinar um sub-grafo conexo de G que contenha os nós de T e cuja soma dos custos dos seus nós seja mínima. Note-se que, dado que uma solução tem que incluir todos os terminais, assume-se sem perda de generalidade, que estes têm custo zero.

O grafo pode ser transformado num grafo orientado substituindo cada aresta $[u, v]$ por um arco (u, v) com custo $c(v)$ e um arco (v, u) com custo $c(u)$. Fixando um dos terminais como origem, o problema é agora determinar uma arborescência mínima que contenha todos os terminais. Problemas em grafos orientados são em geral mais complexos do que em grafos não orientados, pelo que se opta geralmente por trabalhar com a versão original.

O problema foi apresentado por Segev [1987], embora tenha tratado apenas um caso particular.

Feige [1998] provou que, a menos que existam algoritmos quase-polinomiais, i.e. algoritmos com tempo de execução $2^{O((\log n)^c)}$ com c constante, para NP, não existe nenhum algoritmo polinomial com um factor de aproximação ao óptimo inferior a $(1 - o(1)) \log(n)$ para o problema do conjunto de cobertura, onde n é o número de elementos do conjunto base. Klein and Ravi [1995] mostraram, citando uma comunicação pessoal de P. Berman de 1991, que o problema do conjunto de cobertura pode ser reduzido ao problema de Steiner em grafos com custos nos nós, através de uma redução que preserva o factor de aproximação. Este resultado implica que não existe algoritmo polinomial com um factor de aproximação inferior a $(1 - o(1)) \log(p)$, onde p é o número de terminais, para a versão com custos nos nós, a não ser que se restrinja a classe de grafos.

Numa revisão sobre o estado da arte do problema de Steiner em grafos feita em 1987, Winter [1987] enumera sete algoritmos exactos (não polinomiais) e seis heurísticas com tempos de execução polinomial, todos para a versão de custos nas arestas. A última frase do artigo é "Further investigation of the vertex-weighted SPN is needed". De facto, comparado com o caso anterior a literatura para este caso é escassa.

A primeira heurística, com um RD provado, de $2 \ln p$ foi publicada em 1995 por Klein and Ravi [1995]. Trata-se de uma adaptação do algoritmo de Rayward-Smith, descrito acima, ao problema com custos nos nós. A heurística mantém um conjunto de árvores disjuntas que no inicio contêm

apenas cada um dos terminais. A estratégia seguida é a junção iterativa das árvores até que exista apenas uma. Em cada iteração é seleccionado o nó v e um conjunto não singular de árvores por forma a minimizar (custo do nó v mais a soma da distância às árvores)/número de árvores. As duas árvores com menor distância ao nó v seleccionado são então ligadas, diminuindo o número de árvores de uma unidade em cada iteração. No mesmo artigo é apresentada uma variante do algoritmo para a versão nos nós da formulação genérica de Goemans and Williamson [1995] (ver 2.3.1) com o mesmo RD. Esta variante permite obter heurísticas para o vasto conjunto de problemas formulado por Goemans e Williamson.

Guha and Khuller [1999] melhoraram o RD para $1,35 \ln p$ generalizando o conceito de *aranhas*, árvores em que no máximo um dos nós tem grau superior a 2, utilizado por Klein e Ravi na prova do RD de $2 \ln p$. O algoritmo é, no entanto, demasiado lento para aplicar em problemas práticos. No mesmo artigo apresentaram uma versão mais simples com um RD de $1,61 \ln p$. Tanto quanto sabemos, estas continuam a ser as heurísticas com melhor factor de aproximação ao óptimo no pior caso.

Para alguns tipos de grafos, com estrutura particular, existem algoritmos com RD constante. Em 2009 Demaine et al. [2009] publicaram um algoritmo utilizando também a metodologia de Goemans e Williamson com uma formulação de cobertura por cortes do problema de Steiner com custos nos nós. Trata-se de um algoritmo primal-dual que usa a relaxação linear do dual da formulação para juntar as árvores que inicialmente são compostas por cada um dos terminais. Em cada iteração é adicionado o nó correspondente à restrição do dual que fica saturada incrementando uniformemente as variáveis duais. O algoritmo tem um RD de 6 para o caso de grafos planares.

Para o caso de grafos em que os nós estão sobre um plano e existe uma aresta entre dois nós se a distância euclidiana entre eles for menor ou igual que 1 ("Unit Disk Graphs (UDG)"), existem resultados interessantes. Zou et al. [2008] apresentaram uma redução, para este caso, de grafos com custos nos nós ao problema com custos nas arestas e mostraram que qualquer ρ-aproximação a este conduz a um RD de $2,5\rho$ para o problema com custo nos nós. Como o melhor RD conhecido é de 1,39 temos um RD de 3,48 para este caso.

Em 2010 Xu et al. [2010] adaptaram os algoritmos de Berman-Ramayer [Berman and Ramaiyer, 1994] e Zelikovsky (1993 e 1996) [Zelikovsky, 1993, 1996] para o caso de grafos com custos nos nós restringidos a UDG. Obtiveram aproximações de 3.93, 4.33 e 2.61 respectivamente.

2.3 Floresta de Steiner

Dado um grafo não orientado $G = (V, E)$ e um conjunto de subconjuntos disjuntos de V, $T^1, T^2, \cdots, T^\tau$, pretende-se determinar um sub-grafo de G com custo mínimo, em que cada par de nós de cada T^k estão ligados.

2.3.1 Custos nas arestas

Formulações

O artigo já citado de Magnanti and Raghavan [2005] trata explicitamente o caso da floresta de Steiner com custos nas arestas. A formulação com cortes assume neste caso a forma:

$$\min \sum_{\{i,j\} \in E} c_{ij} x_{ij} \tag{2.11a}$$

sujeito a:

$$\sum_{\{i,j\} \in \delta(S)} x_{ij} \geq 1, \quad \begin{array}{l} \forall S \,:\, \emptyset \subset S \subset V, \\ S \cap T_i \neq \emptyset, \\ (V \setminus S) \cap T_i \neq \emptyset \text{ para algum } i \end{array} \tag{2.11b}$$

$$x_{ij} \in \{0, 1\} \quad \forall \{i, j\} \in E \tag{2.11c}$$

Introduziram também um processo de orientação das arestas que conduziu a uma formulação com fluxos, mais forte do que a formulação por cortes. Esta formulação é no entanto muito complexa e com um número muito grande de restrições, embora polinomial, o que faz com que a sua utilização em problemas de grande dimensão seja impraticável.

Heurísticas

O primeiro algoritmo [Goemans and Williamson, 1997] para este problema deve-se a Agrawal, Klein, and Ravi [Agrawal et al., 1995] e tem um RD de 2. Este trabalho motivou o de Goemans e Williamson [Goemans and

Williamson, 1995] que apresentaram uma técnica genérica baseada numa formulação de programação inteira:

$$\begin{array}{ll} \min \sum_{e \in E} c_e x_e & \\ \text{s.a} & \\ x(\delta(S)) \geq f(S) & \emptyset \neq S \subset V \\ x_e \in \{0,1\} & e \in E \end{array}$$

onde $\delta(S)$ designa o conjunto de arestas com um único nó em S e $x(F) = \sum_{e \in F} x_e$. A formulação pode ser interpretada como um problema de cobertura em que se pretende determinar um conjunto de arestas com custo mínimo que cubram todos os cortes $\delta(S)$ correspondentes a conjuntos S para os quais $f(S) = 1$. Os autores designam a função $f : 2^V \rightarrow \{0,1\}$ por função própria quando verifica as seguintes propriedades:

1. Simetria. $f(S) = f(V \setminus S) \ \ \forall S \subseteq V$
2. Disjunção. Se $A \cap B = \emptyset$ então $f(A) = f(B) = 0 \Rightarrow f(A \cup B) = 0$

Goemans e Williamson apresentam um algoritmo primal-dual genérico que para o caso da floresta de Steiner se reduz ao de Agrawal et al.

Diferentes definições para a função $f(S)$ modelam distintos problemas incluindo o da floresta de Steiner. Vários autores têm utilizado esta técnica para criar algoritmos para diversos problemas, incluindo uma das formulações apresentadas nesta dissertação.

Para o caso da floresta de Steiner com custos nas arestas apresenta-se o algoritmo, seguindo o trabalho de Vazirani [2003], dada a sua importância para a formulação e heurística que apresentamos em 3.2 e 4.1 respectivamente.

Seja

$$f(S) = \begin{cases} 1 \text{ se para algum } k, S \text{ inclui algum terminal de } T^k \text{ mas não todos} \\ 0 \text{ caso contrário} \end{cases}$$

$(S, \overline{S})$ representa um corte de G e $f : 2^V \rightarrow \{0,1\}$ é uma função definida sobre o conjunto de todos os subconjuntos de V. Introduzindo uma variável x_e para todos os $e \in E$ que assume o valor 1 se a aresta e for escolhida para a solução e 0 no caso contrário a formulação do problema é:

$$\min \sum_{e \in E} c_e x_e \tag{2.12}$$

sujeito a:

$$\sum_{e:e\in\delta(S)} x_e \geq f(S), \quad S \subseteq V \tag{2.13}$$

$$x_e \in \{0,1\}, \quad e \in E \tag{2.14}$$

onde $\delta(S)$ representa o conjunto de arestas que atravessam o corte $(S, \overline{S})$.

A relaxação linear deste problema é:

$$\min \sum_{e\in E} c_e x_e \tag{2.15}$$

sujeito a:

$$\sum_{e:e\in\delta(S)} x_e \geq f(S), \quad S \subseteq V \tag{2.16}$$

$$x_e \geq 0, \quad e \in E \tag{2.17}$$

omite-se a restrição $x_e \leq 1$ por ser redundante.

O dual da relaxação linear é:

$$\max \sum_{S\subseteq V} f(S) y_S \tag{2.18}$$

sujeito a:

$$\sum_{S:e\in\delta(S)} y_S \leq c_e, \quad e \in E \tag{2.19}$$

$$y_S \geq 0, \quad S \subseteq V \tag{2.20}$$

Um algoritmo primal-dual exacto actua sobre as condições de complementaridade, escolhendo em cada iteração uma não satisfeita e alterando as soluções do dual e do primal por forma a satisfazerem a condição. A heurística que se descreve para obter soluções aproximadas, utiliza o conceito de esquema primal-dual com sincronização, fazendo crescer as variáveis duais de forma sincronizada. O algoritmo não tenta satisfazer uma condição de complementaridade específica, em vez disso tenta várias possibilidades e

escolhe a que conduz a melhorias no dual. O algoritmo termina quando a solução do primal é admissível.

A principal modificação deste tipo de algoritmos primal-dual é que enquanto os algoritmos exactos impõem as condições de complementaridade do primal e do dual, estes impõem as condições do primal mas relaxam as do dual [Goemans and Williamson, 1997].

Neste caso, a condição de complementaridade primal é para cada $e \in E$, $x_e \neq 0 \Rightarrow \sum_{S:e\in\delta(S)} y_S = c_e$, ou seja todas as arestas escolhidas têm que saturar a restrição.

O algoritmo começa com todas as variáveis primais e duais a zero. A solução primal corrente indica quais os cortes que devem ser *levantados*, i.e. quais as variáveis y_S para as quais se deve ter $y_S > 0$. Por sua vez a solução corrente do dual indica quais as arestas que devem ser escolhidas, pelo que o algoritmo melhora iterativamente a admissibilidade do primal e a optimalidade do dual até que seja obtida uma solução primal admissível.

O conjunto S diz-se activo se é uma componente conexa na solução corrente e $f(S) = 1$. Em cada iteração as variáveis duais dos conjuntos activos são aumentadas de forma sincronizada até que alguma das restrições (2.19) seja saturada, escolhendo-se a aresta correspondente.

O algoritmo é:

1. $F \leftarrow \emptyset$
2. $y_S \leftarrow 0 \ \ \forall S \subseteq V$
3. Enquanto a solução F não for primal admissível
   (a) aumentar simultaneamente todos os y_S correspondentes a conjuntos S activos até que alguma das restrições (2.19) seja saturada. Escolher o e correspondente.
   (b) $F \leftarrow F \cup \{e\}$
4. Podar a solução fazendo $F' = \{e \in F | F \setminus \{e\}$ é primal não admissível$\}$
5. Devolver F'.

O algoritmo descrito tem um factor de aproximação ao óptimo de 2 [Vazirani, 2003].

2.3.2 Custos nos nós

Os trabalhos publicados sobre este caso são escassos, resumindo-se quase sempre a notas afirmando que é possível adaptar os algoritmos ao caso de custos nos nós.

Klein and Ravi [1995] mostram como adaptar o seu algoritmo, derivado do de Rayward-Smith, para os problemas cobertos pela formulação genérica de Goemans and Williamson [1995] e portanto também para o caso da floresta de Steiner com custos nos nós. Para tal basta modificar a função $f(v)$ de Rayward-Smith para:

$$\frac{\text{custo do nó + soma das distâncias às árvores activas}}{\text{número de árvores activas}}$$

O factor de aproximação do algoritmo é $2\log(p)$ onde p é o número de terminais.

Demaine et al. [2009] mostram que o seu algoritmo pode ser também utilizado para o caso da floresta de Steiner, mantendo um RD de 6 no caso de grafos planares.

Capítulo 3

Formulações

3.1 Formulações com fluxos

Formulações com recurso a fluxos têm sido usadas para representar formalmente problemas de Steiner. Veja-se por exemplo Wong [1984] para o caso do problema da árvore de Steiner com custos nas arestas, Segev [1987] para um caso particular do problema da árvore de Steiner com pesos nos nós e Magnanti and Raghavan [2005] para vários problemas de conexidade em redes, incluindo o da floresta de Steiner com custos nas arestas.

Apresenta-se uma formulação para o PLMT baseada em fluxos multiproduto.

Sejam:

$w_v \geq 0$ um custo associado a cada nó v do grafo $G = (V, E)$, $w_v = 0$ se $v \in T^k$ para algum $k = 1, \cdots, \tau$, i.e. os nós terminais têm custo 0.

$p_k = \left|T^k\right|,\ k = 1, \cdots, \tau$ o número de terminais do tipo k.

$t_1^k \in T^k,\ k = 1, \cdots, \tau$ um nó escolhido arbitrariamente para origem do fluxo para cada um dos τ tipos de terminais.

Considerem-se variáveis f_{uv}^k representando a quantidade de fluxo do tipo k no arco entre os nós u e v, e variáveis $x_v \in \{0, 1\}$ indicando se o nó v é seleccionado para ligar terminais ($x_v = 1$) ou não ($x_v = 0$).

Com as variáveis **f** e **x**, PLMT pode ser formulado da forma seguinte:

$PLMT_1$:

$$\min \sum_{v \in V} w_v x_v \tag{3.1a}$$

sujeito a:

$$\sum_{u \in V^k} f_{uv}^k - \sum_{w \in V^k} f_{vw}^k = \begin{cases} -(p_k - 1) & \text{se } v = t_1^k \\ 1 & \text{se } v \in T^k \setminus \{t_1^k\}, \\ 0 & \text{se } v \in V^k \setminus T^k \end{cases} \quad k = 1, \cdots, \tau \tag{3.1b}$$

$$\sum_{u \in V^k} f_{uv}^k \leq (p_k - 1)\, x_v, \qquad \begin{array}{l} v \in V^k \setminus \left\{t_1^k\right\}, \\ k = 1, \cdots, \tau \end{array} \tag{3.1c}$$

$$x_{t_1^k} = 1 \qquad k = 1, \cdots, \tau \tag{3.1d}$$

$$x_v \in \{0, 1\}, \qquad v \in V \tag{3.1e}$$

$$f_{uv}^k \geq 0, \;\; u, v \in V^k, \;\; k = 1, \cdots, \tau. \tag{3.1f}$$

As equações de equilíbrio (3.1b) garantem a conexidade dos terminais de T^k dado que um fluxo de $p_k - 1$ unidades é enviado pelo terminal seleccionado para origem e uma unidade de fluxo é consumida por cada um dos restantes $p_k - 1$ terminais. As restrições de capacidade (3.1c) garantem que o fluxo atravessa os nós seleccionados.

Em vez de enviar um fluxo de $p_k - 1$ unidades a partir de cada origem, podemos formular o problema enviando um fluxo unitário a partir de um terminal origem em T^k para cada um dos outros $p_k - 1$ terminais. Sejam $t_1^k, t_2^k, \cdots, t_{p_k}^k$ os elementos do conjunto de terminais T^k e escolha-se para origem t_1^k. Utilizam-se variáveis f_{uv}^{ki} para designar a quantidade de fluxo de tipo k entre os nós u e v com origem em t_1^k e destino t_i^k $(i = 2, \cdots, p_k)$.

PLMT pode agora ser formulado por:

$PLMT_2$:

$$\min \sum_{v \in V} w_v x_v \tag{3.2a}$$

sujeito a:

$$\sum_{u\in V^k} f^{ki}_{uv} - \sum_{w\in V^k} f^{ki}_{vw} = \begin{cases} -1 & \text{se } v = t^k_1 \\ 1 & \text{se } v = t^k_i \\ 0 & \text{se } v \in V^k \setminus T^k \end{cases}, \quad \begin{matrix} k = 1, \cdots, \tau \\ i = 2, \cdots, p_k \end{matrix} \tag{3.2b}$$

$$\sum_{u\in V^k} f^{ki}_{uv} \leq x_v, \quad \begin{matrix} v \in V^k, \\ k = 1, \cdots, \tau \\ i = 2, \cdots, p_k \end{matrix} \tag{3.2c}$$

$$\sum_{u\in V^k} f^{ki}_{ut^k_1} = 0, \quad \begin{matrix} k = 1, \cdots, \tau \\ i = 2, \cdots, p_k \end{matrix} \tag{3.2d}$$

$$\sum_{u\in V^k} f^{ki}_{t^k_i u} = 0, \quad \begin{matrix} k = 1, \cdots, \tau \\ i = 2, \cdots, p_k \end{matrix} \tag{3.2e}$$

$$x_{t^k_1} = 1 \qquad k = 1, \cdots, \tau \tag{3.2f}$$

$$x_v \in \{0, 1\}, \qquad v \in V \tag{3.2g}$$

$$f^{ki}_{uv} \geq 0, \;\; u, v \in V^k, \;\; k = 1, \cdots, \tau, \; i = 2, \cdots, p_k. \tag{3.2h}$$

As equações (3.2b), (3.2c) e (3.2f) garantem que todos os terminais pertencem à solução e que para cada tipo k pertencem à mesma componente conexa do grafo $\langle V^k \rangle$. As equações (3.2d)-(3.2e) eliminam algumas das variáveis de fluxo (estão implícitas mas a experiência mostrou que a sua inclusão na formulação diminui o tempo necessário para obter a solução da relaxação linear do problema).

Comparamos em seguida as duas formulações.

Definição 3.1.1 *Dado um poliedro $Q \subseteq (R^n \times R^p)$, a projecção de Q sobre o sub-espaço R^n, representa-se por $proj_x(Q)$ e é definida por:*

$$proj_x(Q) = \{x \in R^n \; : \; (x, w) \in Q \text{ para algum } w \in R^p\}.$$

Designem-se por P_1 e P_2 os poliedros obtidos a partir das relaxações lineares das restrições (3.1b)-(3.1f) e (3.2b)-(3.2h) respectivamente.

As duas formulações têm um conjunto de variáveis comum, x, e variáveis diferentes para representar os fluxos.

A proposição seguinte estabelece a comparação entre as duas formulações.

Proposição 3.1.2 *$PLMT_2$ é uma formulação mais forte do que $PLMT_1$, i.e. $proj_x(P_2) \subseteq proj_x(P_1)$.*

Dem: Considere-se uma solução admissível $(\mathbf{x}, \mathbf{f})$ da relaxação linear de $PLMT_2$. Por (3.2c)

$$\sum_{u \in V^k} f_{uv}^{ki} \leq x_v$$

somando em i

$$\sum_{i=2}^{p_k} \sum_{u \in V^k} f_{uv}^{ki} \leq (p_k - 1)x_v.$$

Tomando $f_{uv}^k = \sum_{i=2}^{p_k} f_{uv}^{ki}$ tem-se:

$$\sum_{u \in V^k} f_{uv}^k \leq (p_k - 1)x_v$$

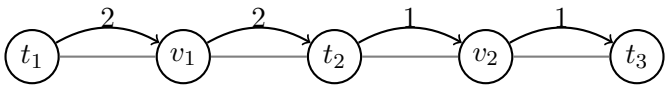
e portanto qualquer solução admissível $\mathbf{x}$ da relaxação linear de $PLMT_2$ é também admissível para a relaxação de $PLMT_1$, o que implica necessariamente que:

$$proj_x(P_2) \subseteq proj_x(P_1).$$

Para provar que existem instâncias para as quais $proj_x(P_2) \subset proj_x(P_1)$ basta mostrar que existe um ponto de $proj_x(P_1)$ que não está em $proj_x(P_2)$.

Considere-se o grafo da Fig. 3.1 onde $\tau = 1$ e $p_1 = 3$

Figura 3.1: Solução da relaxação linear P_1



Os arcos representam os fluxos na solução óptima da relaxação linear $PLMT_1$. A solução óptima é $(x_{t_1}, x_{v_1}, x_{t_2}, x_{v_2}, x_{t_3}) = \left(1, 1, 1, \frac{1}{2}, 1\right)$ que não é um ponto de P_2 dado que por (3.2b) e (3.2f) temos que ter $f^{13}_{t_2v_2} = 1$ e portanto $x_{v_2} = \frac{1}{2}$ viola (3.2c).

Isto prova que $PLMT_2$ é uma formulação mais forte do que $PLMT_1$. □

A formulação compacta (3.2a)-(3.2h) obtida acima, será usada para obter soluções exactas para o PLMT (ou enquadramentos para o óptimo quando tal não for possível) para instâncias de dimensão moderada, por forma a aferir a qualidade das soluções obtidas com as heurísticas do capítulo 4.

3.1.1 Comparação empírica

Para comparar as relaxações lineares das duas formulações utilizou-se um subconjunto dos grafos de teste descritos em 5.1.2. Para cada $|V|$, $|E|$ e τ existem 10 problemas. Foram usados apenas aqueles para os quais se obteve pelo menos duas soluções óptimas, usando o CPLEX limitado a uma hora de execução. As características da máquina utilizada estão descritas na secção referida.

Os resultados estão descritos na Tabela 3.1. Os valores apresentados são as médias dos *np* problemas para cada tipo de grafo. LP/IP é a razão entre a solução da relaxação linear e o valor do óptimo para o problema com as variáveis inteiras. As quatro últimas colunas apresentam o tempo médio de execução para obter os óptimos da relaxação linear e do problema inteiro respectivamente. Os valores omissos significam que não foi possível obter o óptimo, no tempo fixado, para um ou mais dos problemas apresentados. Nunca se deu o caso de ser possível obter o óptimo com a formulação $PLMT_1$ mas não com a $PLMT_2$.

Os resultados mostram que, para estes problemas, a formulação $PLMT_2$ é bastante mais forte, sendo a média do rácio LP/IP superior a 90% excepto em dois casos enquanto que $PLMT_1$ apenas ultrapassa os 85% em dois casos.

Os tempos de execução para obtenção do óptimo da relaxação linear, reflectem o maior número de variáveis e restrições de $PLMT_2$ (ver Tabela 3.2) sendo bastante mais elevados. Como era de esperar, para os mesmos valores de $|V|$ e $|E|$ os tempos da relaxação linear aumentam com τ.

Tabela 3.1: Comparação das formulações com fluxos.

$\|V\|$	$\|E\|$	τ	np	LP/IP		LP (segs.)		IP (segs.)	
				$PLMT_1$	$PLMT_2$	$PLMT_1$	$PLMT_2$	$PLMT_1$	$PLMT_2$
100	342	1	4	0.773	1.000	0.00	0.00	0.03	0.02
		2	8	0.765	0.996	0.00	0.01	0.08	0.03
		3	10	0.765	0.984	0.00	0.01	0.08	0.07
		4	7	0.673	0.990	0.01	0.03	0.13	0.08
		5	5	0.764	0.988	0.01	0.03	0.22	0.08
		6	5	0.832	0.985	0.01	0.04	0.14	0.12
		7	3	0.653	1.000	0.01	0.02	0.05	0.06
		8	7	0.770	0.991	0.02	0.07	0.14	0.14
		10	9	0.736	0.992	0.03	0.07	0.19	0.16
400	1482	1	11	0.611	0.946	0.01	0.08	3.75	0.67
		2	13	0.799	0.970	0.04	0.24	3.51	1.48
		3	6	0.642	0.892	0.08	0.91	21.78	10.85
		4	9	0.750	0.919	0.28	1.54	32.65	23.62
		5	4	0.694	0.918	0.21	1.17	667.60	491.82
		6	7	0.690	0.920	0.58	4.30	248.35	57.54
		7	2	0.712	0.879	0.87	7.72	258.11	259.71
		8	4	0.712	0.931	0.96	6.55	349.63	119.87
		10	2	0.733	0.972	0.80	7.20	19.31	18.14
900	3422	1	13	0.837	0.971	0.05	0.21	10.20	1.39
		2	16	0.802	0.928	0.40	2.31		225.06
		3	8	0.725	0.901	0.67	2.09		364.26
		4	5	0.672	0.907	0.98	3.55		706.50
		5	7	0.682	0.906	0.95	4.55		200.09
		7	3	0.767	0.937	1.78	8.68	1345.97	342.49
1600	6162	1	18	0.756	0.963	0.09	0.78		187.87
		2	9	0.824	0.961	0.52	1.31	159.37	16.95
		3	8	0.729	0.925	1.27	5.05		927.20
		4	2	0.859	0.933	2.83	5.04		484.68
		5	2	0.823	0.944	1.02	1.72		1178.29
2500	9702	1	22	0.854	0.977	0.25	0.70		143.83
		2	9	0.737	0.931	1.13	3.37		432.69
		3	4	0.676	0.956	1.68	10.12		655.07

Tabela 3.2: Formulações com fluxos: número de variáveis e restrições.

	$PLMT_1$	$PLMT_2$
Variáveis:	$\|V\| + \sum_{k=1}^{\tau} \left\|E^k\right\|$	$\|V\| + \sum_{k=1}^{\tau} (p_k - 1) \left\|E^k\right\|$
Restrições:	$\tau + 2\sum_{k=1}^{\tau} \left\|V^k\right\|$	$\tau + 2\sum_{k=1}^{\tau} (p_k - 1) \left(\left\|V^k\right\| + 1\right)$

3.2 Formulação por cortes

Pode escrever-se uma formulação para o PLMT baseada em cortes de cobertura que é uma extensão, adaptada para pesos nos nós, da formulação de Vazirani [2003] para o problema com pesos nas arestas. Esta última é, por sua vez, uma aplicação do problema genérico proposto por Goemans and Williamson [1995] para o desenho de problemas em redes com custos nas arestas. Uma formulação semelhante foi dada por Demaine et al. [2009] para o problema da floresta de Steiner com pesos nos nós.

Para cada $k = 1, 2, \ldots, \tau$, considere-se a função $f^k : 2^{V^k} \to \{0, 1\}$ definida por $f^k(S) = 1$ se e só se $\emptyset \neq S \cap T^k \neq T^k$ (i.e., S inclui pelo menos um terminal de T^k, mas não todos).

Usando as variáveis x_v, previamente definidas, PLMT pode ser formulado como:

$$\min \sum_{v \in V} w_v x_v \tag{3.3a}$$

sujeito a:

$$\sum_{v \in \Gamma^k(S)} x_v \geq f^k(S), \qquad \begin{array}{l} S \subseteq V^k \\ k = 1, \cdots, \tau \end{array} \tag{3.3b}$$

$$x_v \in \{0, 1\}, \qquad v \in V \tag{3.3c}$$

onde $\Gamma^k(S)$ representa o conjunto de nós $v \in V^k \setminus S$ adjacente a pelo menos um nó em S.

Proposição 3.2.1 *A solução deste problema de programação inteira é uma solução do PLMT.*

Dem: Suponha-se por contradição que algum par de terminais do tipo k, (s^k, t^k), não é ligado por nós para os quais se tem $x_v = 1$. Seja S o conjunto formado por s^k e pelos nós a ele ligados na solução. Por hipótese tem-se $f^k(S) = 1$ dado que S contem s^k mas não t^k. Por outro lado a nossa escolha de S implica que para todo o $v \in \Gamma^k(S)$ se tem $x_v = 0$, o que viola 3.3b. □

Contrariamente à formulação com fluxos, que tem um número polinomial de restrições, a formulação acima tem um número exponencial de desigualdades. A formulação (3.3) será usada no Capítulo 4, para obter um procedimento heurístico primal-dual para o PLMT.

Capítulo 4

Heurísticas

4.1 Heurística primal-dual

4.1.1 O método primal-dual

O método primal-dual foi proposto por Dantzig et al. [1956] para resolver problemas de programação linear. Considere-se o par de problemas duais:

$$\min c^T x$$

sujeito a:

$$\begin{aligned} Ax &\geq b \\ x &\geq 0 \end{aligned}$$

e

$$\max b^T y$$

sujeito a:

$$\begin{aligned} A^T y &\leq c \\ y &\geq 0. \end{aligned}$$

As relações de complementaridade primais são:

$$x_j > 0 \Rightarrow A^j y = c_j$$

enquanto que as relações de complementaridade duais são:

$$y_i > 0 \Rightarrow A_i x = b_i$$

onde A^j representa a j-ésima coluna de de A e A_i a i-ésima linha.

Dada uma solução y, admissível para o dual, o problema de encontrar uma solução x admissível para o primal que respeite as condições de complementaridade leva-nos a outro problema de optimização: encontrar x por forma a minimizar a violação das restrições do primal e das condições de complementaridade. Este problema pode ser formulado de várias formas. Segue-se o trabalho de Goemans and Williamson [1997] sobre o método primal-dual e respectivas heurísticas.

$$Z_{inf} = \min \sum_{i \notin I} s_i + \sum_{j \notin J} x_j$$

sujeito a:

$$\begin{array}{ll} A_i x \geq b_i & i \in I \\ A_i x - s_i = b_i & i \notin I \\ x \geq 0 & \\ s \geq 0 & \end{array}$$

com $I = \{i : y_i = 0\}$ e $J = \{j : A^j y = c_j\}$.

Se $Z_{inf} = 0$ então x e y são soluções óptimas para o primal e dual respectivamente. Caso contrário considere-se o dual deste problema:

$$\max b^T y^1$$

sujeito a:

$$\begin{array}{ll} A^j y^1 \leq 0 & j \in J \\ A^j y^1 \leq 1 & j \notin J \\ y_i^1 \geq -1 & i \notin I \\ y_i^1 \geq 0 & i \in I \end{array}$$

$Z_{inf} > 0$ implica que $b^T y^1 > 0$. É possível encontrar um $\epsilon > 0$ tal que $y^2 = y + \epsilon y^1$ é uma solução admissível para o dual [Goemans and Williamson, 1997], ou seja, se não for possível determinar um x que respeite as condições de complementaridade podemos determinar uma solução dual admissível y^2 tal que $b^T y^2 = b^T y + \epsilon b^T y^1 > b^T y$ e portanto com melhor valor para a função objectivo.

O método transforma a solução de um problema de programação linear na solução de uma sequência de problemas. Embora possa parecer que não há vantagem nesta transformação deve notar-se que o vector c não aparece no problema de minimização das violações (nem no seu dual). Em problemas de redes, este vector corresponde aos custos das arestas ou nós e portanto o método reduz o problema com pesos a um problema sem estes tornando-o, em geral, mais fácil de resolver [Goemans and Williamson, 1997].

Muitos problemas de optimização combinatória podem ser formulados utilizando programação inteira. Em alguns casos, a relaxação linear destes problemas tem uma matriz A que é totalmente unimodular (i.e. qualquer sub-matriz quadrada de A tem determinante -1, 0 ou 1) o que implica que a solução da relaxação linear é inteira. Entre estes, estão o da determinação do caminho mais curto entre dois nós ou o do fluxo máximo numa rede. Na maioria dos casos isto não acontece e não é possível recorrer à programação linear para resolver o problema.

Para obter soluções aproximadas para um problema impomos as condições de complementaridade primais mas relaxamos as do dual. Em cada iteração mantém-se uma solução admissível para o dual e as variáveis do primal associadas às restrições duais saturadas são básicas. Se a solução do primal não for admissível então podemos aumentar o valor da solução do dual. O processo termina quando se obtém uma solução admissível para o primal.

4.1.2 Heurística primal-dual para o PLMT

Uma heurística primal-dual baseada numa formulação de cortes foi proposta inicialmente por Goemans and Williamson [1995] para o caso de pesos nas arestas. A heurística foi posteriormente adaptada para o caso de pesos nos nós por Demaine et al. [2009]. Apresenta-se uma extensão destas, adiante designada por PD, para o PLMT.

A relaxação linear da formulação (3.3) é:

$$\min \sum_{v \in V} w_v x_v \tag{4.1}$$

sujeito a:

$$\sum_{v \in \Gamma^k(S)} x_v \geq f^k(S), \qquad \begin{matrix} S \subseteq V^k \\ k = 1, \cdots, \tau \end{matrix}$$

$$x_v \geq 0, \qquad v \in V$$

omitiram-se as restrições $x_v \leq 1$ por serem redundantes.

O dual é:

$$\max \sum_{k=1}^{\tau} \sum_{S \subseteq V^k} f^k(S) y^k(S)$$

sujeito a:

$$\sum_{k=1}^{\tau} \sum_{S \subseteq V^k : v \in \Gamma^k(S)} y^k(S) \leq w_v \quad v \in V \tag{4.2}$$

$$y^k(S) \geq 0 \qquad \begin{matrix} S \subseteq V^k \\ k = 1, \cdots, \tau \end{matrix} .$$

O dual tem uma variável $y^k(S)$ para cada $S \subseteq V^k$, contudo só temos que nos preocupar com os $y^k(S)$ para os quais $f^k(S) = 1$.

Sejam X um subconjunto de nós de V contendo pelo menos todos os terminais, $X^k = X \cap V^k$ e $\mathcal{C}(\langle X^k \rangle)$ as componentes conexas de $\langle X^k \rangle$. X é uma solução admissível se e só se $f^k(C) = 0 \ \ \forall C \in \mathcal{C}(\langle X^k \rangle), \ \ k = 1, 2, \cdots, \tau$.

O algoritmo começa com todos os terminais em X e $y^k(S) = 0$ para todo o k e S. Em cada iteração calculam-se as componentes conexas $C \in \mathcal{C}(\langle X^k \rangle)$. Se para algum C se tiver $f^k(C) = 1$ dizemos que a componente

conexa é violada. Incrementamos o valor de $y^k(C)$, $\forall C \in \mathcal{C}(\langle X^k \rangle)$ pela mesma quantidade até que uma das restrições (4.2) seja saturada. O nó correspondente v é acrescentado a X. Se não existirem componentes conexas violadas o processo termina com uma solução do problema. Na fase seguinte removemos nós, pela ordem inversa pela qual foram adicionados, que não sejam necessários para manter a solução primal admissível.

Figura 4.1: Heurística PD

1. $X \leftarrow \bigcup_k T^k$
2. Para $k = 1, \cdots, \tau$ calcular componentes conexas $\mathcal{C}(\langle X^k \rangle)$. $y^k(C) \leftarrow 0$ para todo o $C \in \mathcal{C}(\langle X^k \rangle)$
3. Enquanto X não for $PLMT$ admissível
   (a) Incrementar $y^k(C)$ simultaneamente até que para algum $v \in V \setminus X$, uma das restrições (4.2) seja saturada.
   (b) $X \leftarrow X \cup \{v\}$
   (c) Para $k = 1, \cdots, \tau$ recalcular X^k e $\mathcal{C}(\langle X^k \rangle)$.
4. Tornar X minimal. Para todo o $v \in X \setminus \bigcup_k T^k$, pela ordem inversa pela qual foram adicionados, remover v de X se $X \setminus \{v\}$ for $PLMT$ admissível
5. Devolver X.

A complexidade do algoritmo não pode ser expressa de forma clara. No passo (3c), (ver figura 4.1), é necessário calcular as componentes conexas em τ grafos que têm na iteração i, no máximo $p_k + i$ nós e um número indeterminado de arestas. O mesmo se passa no passo (4). Se o algoritmo necessitar de muitas iterações para encontrar uma solução, então a dimensão dos grafos $\langle X^k \rangle$ será grande e a heurística poderá ser impraticável. No Capítulo 5 analisaremos o desempenho prático da heurística.

O algoritmo é semelhante a um dos algoritmos propostos por Goemans and Williamson [1997] para problemas com custos nas arestas. Para obter uma α-aproximação para a heurística segue-se um processo semelhante ao utilizado para provar o teorema 4.2 do trabalho citado.

Designe-se por X_f a solução final obtida com o algoritmo descrito na

Figura 4.1. O custo da solução é:

$$C(X_f) = \sum_{v \in X_f} w_v$$

Como em cada iteração, da fase 3, se junta à solução o nó v que satura a restrição 4.2 tem-se:

$$w_v = \sum_{k=1}^{\tau} \sum_{S \subseteq V^k : v \in \Gamma^k(S)} y^k(S)$$

e o custo do primal pode se expresso por:

$$C(X_f) = \sum_{k=1}^{\tau} \sum_{v \in X_f} \sum_{S \subseteq V^k : v \in \Gamma^k(S)} y^k(S).$$

Os dois somatórios da direita somam $y^k(S)$ tantas vezes quantas o número de nós de X_f adjacentes a S ou seja $\left|X_f \cap \Gamma^k(S)\right|$. Tem-se assim:

$$C(X_f) = \sum_{k=1}^{\tau} \sum_{S \subseteq V^k} \left|X_f \cap \Gamma^k(S)\right| y^k(S). \tag{4.3}$$

O custo da solução dual admissível

$$\sum_{k=1}^{\tau} \sum_{S \subseteq V^k} y^k(S)$$

é um limite inferior para o custo da solução óptima.

O algoritmo incrementa simultaneamente as variáveis $y^k(S)$ pela mesma quantidade ϵ_j na iteração j. Designando por $\mathcal{V}_j^k$ o conjunto das componentes conexas, do tipo k, violadas na iteração j, o valor da variável dual $y^k(S)$ é:

$$y^k(S) = \sum_{j : S \in \mathcal{V}_j^k} \epsilon_j \tag{4.4}$$

substituindo (4.4) em (4.3) obtém-se:

$$C(X_f) = \sum_{k=1}^{\tau} \sum_{S \subseteq V^k : y^k(S) > 0} \left| X_f \cap \Gamma^k(S) \right| \sum_{j : S \in \mathcal{V}_j^k} \epsilon_j$$

reordenando os termos do somatório e designando por l o número de iterações efectuadas obtém-se:

$$C(X_f) = \sum_{k=1}^{\tau} \sum_{j=1}^{l} \left(\sum_{S \in \mathcal{V}_j^k} \left| X_f \cap \Gamma^k(S) \right| \right) \epsilon_j. \tag{4.5}$$

Por outro lado o custo do dual na iteração l é expresso por:

$$\sum_{k=1}^{\tau} \sum_{j=1}^{l} \left| \mathcal{V}_j^k \right| \epsilon_j. \tag{4.6}$$

Comparando as expressões (4.5) e (4.6) conclui-se que $C(X_f)$ é no máximo α vezes o custo da solução dual se para todas as iterações $j = 1, \cdots, l$ se tiver:

$$\sum_{S \in \mathcal{V}_j^k} \left| X_f \cap \Gamma^k(S) \right| \leq \alpha \left| \mathcal{V}_j^k \right|.$$

Esta expressão depende da solução final obtida no final da fase 4 do algoritmo descrito na Figura 4.1, tendo pouco interesse prático.

Na fase 4 do algoritmo retiram-se, por ordem inversa de inserção, nós redundantes da solução, obtendo-se a solução final X_f. Sejam v_j o nó inserido na solução na iteração j da fase 3, F o conjunto de nós na solução quando é considerada a remoção de v_j e $S = \{v_1, v_2, \cdots, v_{j-1}\}$. Temos que $F = X_f \cup S$ é um *aumento mínimo admissível* de S, i.e. F é admissível, contém S e $\forall v \in F \backslash S$, $F \backslash \{v\}$ não é admissível. Claramente $\left| X_f \cap \Gamma^k(S) \right| \leq \left| F \cap \Gamma^k(S) \right|$. Pode, assim, substituir-se X_f por um aumento mínimo admissível, F, da solução não admissível na iteração j. O exposto provou o teorema seguinte:

Teorema 4.1.1 *O algoritmo PD obtém uma solução para* (3.3) *cujo custo não excede*

$$\alpha \sum_{k=1}^{\tau} \sum_{S \in V^k} y^k(S) \leq \alpha Opt$$

desde que para qualquer solução não admissível X *e qualquer aumento mínimo admissível* F *de* X *se tenha para todo o* k*:*

$$\sum_{S \in \mathcal{V}^k(X)} \left| F \cap \Gamma^k(S) \right| \leq \alpha \left| \mathcal{V}^k \right| \tag{4.7}$$

onde $\mathcal{V}^k$ *representa o conjunto de componentes conexas de* X *para as quais se tem* $f^k(S) = 1$*.*

Se $\tau = 1$, PD reduz-se à heurística de Demaine et al. [2009] para os problemas da árvore de Steiner com custos nos nós e da floresta de Steiner com custos nos nós. O somatório na desigualdade (4.7) conta o número de adjacências entre $F \setminus X$ e as componentes conexas violadas da solução não admissível X. Se para todas as componentes conexas S todos os nós em $\Gamma(S)$ fossem adjacentes a apenas uma componente conexa então $\sum_{S \in \mathcal{V}} |F \cap \Gamma(S)| \leq 2(|\mathcal{V}| - 1)$ dado que o número de adjacências seria igual ao número de arestas entre as componentes violadas e a árvore de suporte destas. Nos casos em que os nós em $F \cap \Gamma(S)$ são adjacentes a mais do que uma componente violada, estes são contados mais do que uma vez. Para o caso de grafos planares [Demaine et al., 2009] provaram que a desigualdade (4.7) é válida para $\alpha = 6$ quando $\tau = 1$.

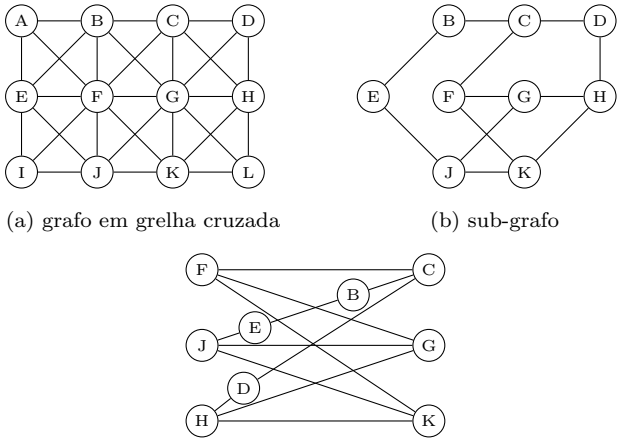
Os grafos que modelam problemas de biologia da conservação têm uma estrutura particular. Associamos células de terreno aos nós e adjacências entre estas às arestas. Normalmente as células são quadrados e consideram-se adjacentes as células que lhe estão fisicamente ligadas. Quando se consideram apenas adjacências no lados dos quadrados estamos perante um *grafo em grelha*. Na investigação que desenvolvemos consideramos também como adjacentes as células ligadas aos vértices dos quadrados.

Definição 4.1.2 *Um grafo em grelha cruzada é uma variante de um grafo em grelha em que cada nó tem no máximo grau 8, 4 nós adjacentes correspondentes aos lados dos quadrados e 4 aos vértices.*

Proposição 4.1.3 *Um grafo* $m \times n$ *em grelha cruzada não é planar para* $m \geq 3$, $n > 3$*.*

Dem: O grafo contém uma sub-divisão de $K_{3,3}$ e portanto, pelo teorema de Kuratowski, não é planar (ver figura 4.2). □

Figura 4.2: Grafo em grelha cruzada



(a) grafo em grelha cruzada

(b) sub-grafo

(c) sub-divisão de $K_{3,3}$

Provaremos em seguida que para o caso de um grafo em grelha cruzada, a heurística PD, com $\tau = 1$, obtém um resultado com uma garantia de aproximação ao óptimo por um factor de 2, para os problemas da árvore e floresta de Steiner com custos nos nós.

No que segue assume-se que F é conexo. Se não for (floresta de Steiner) podem usar-se os mesmo argumentos para cada uma das suas componentes conexas.

Considere-se uma solução parcial não admissível X e o grafo induzido pelo aumento mínimo admissível F. Contraiam-se os nós de cada uma das componentes violadas, S, de X num único nó eliminando as arestas duplicadas. Seja U o conjunto destes nós. Como os conjuntos S são as componentes conexas da solução parcial X, os elementos de U não são adjacentes. Seja F' o conjunto de nós de $F \setminus X$ adjacentes a qualquer $u \in U$, i.e., os elementos de $F \cap \Gamma(S)$ para todo o $S \in \mathcal{V}$.

Lema 4.1.4 *Seja $G = (V, E)$ um grafo em grelha cruzada. O número máximo de adjacências entre qualquer $v \in F'$ e os elementos de U é 4.*

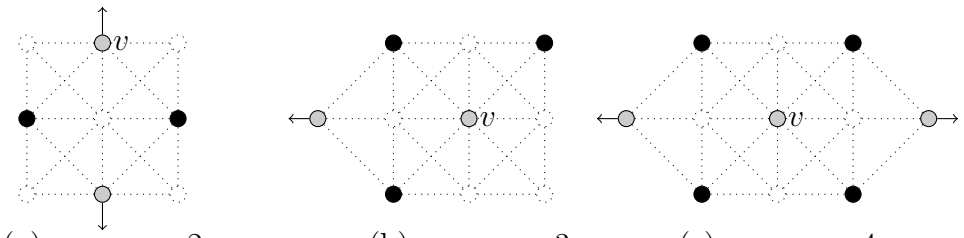
Dem: Como v tem apenas 8 nós adjacentes e os elementos de U não podem ser adjacentes, tem que existir um nó entre qualquer par de nós pertencentes

a U e portanto v não pode ser adjacente a mais de quatro componentes violadas. □

Lema 4.1.5 *Sejam $G = (V, E)$ um grafo em grelha cruzada e $v \in F'$ um nó adjacente a várias componentes cujo conjunto se designa por $\mathcal{S}'$. O contributo dos nós com múltiplas adjacências a estas componentes para a soma $\sum_{S \in \mathcal{S}'} |F \cap \Gamma(S)|$ é no máximo 4, 5 ou 8 para v adjacente a 2, 3 ou 4 componentes respectivamente.*

Dem: A minimalidade (relativamente à inclusão) de F implica este resultado. A Figura 4.3 mostra os casos possíveis de nós de $F \cap \Gamma(S)$ adjacentes a 2, 3 e 4 componentes violadas. Considere-se o caso (c) na Fig. 4.3. Se existisse outro nó adjacente a duas componentes então v seria redundante e teríamos no máximo 3 nós adjacentes a duas componentes cada, mas nenhum adjacente a 4. O mesmo argumento aplica-se aos outros dois casos. □

Figura 4.3: Adjacências múltiplas de F'. Os nós a preto pertencem a componentes violadas e a cinzento são elementos de F'



(a) comum a 2 (b) comum a 3 (c) comum a 4

Podemos agora provar o seguinte teorema:

Teorema 4.1.6 *Seja $G = (V, E)$ um grafo em grelha cruzada, isto é, um grafo em grelha em que cada nó tem no máximo grau 8 (4 nos lados do quadrado e 4 nos vértices), então a heurística PD com $\tau = 1$ obtém um resultado com uma garantia de aproximação ao óptimo por um factor de 2 para os problemas da árvore e floresta de Steiner com custos nos nós.*

Dem: Considere-se a solução parcial X e o aumento mínimo admissível F. Seja G' o grafo induzido por F em que se contraíram as componentes violadas S de X. Considerem-se os sub-grafos de G' nas condições do lema 4.1.5 e contraia-se cada um destes num único nó (ver figura 4.3). O grafo

resultante, G'', tem como terminais, T', as componentes violadas sem adjacências múltiplas mais os nós correspondentes aos sub-grafos contraídos. Seja $F'' = F' \setminus T'$. Por construção nenhum nó de F'' é adjacente a mais do que uma componente violada (caso contrário existiriam nós redundantes em T' e F não seria um aumento mínimo).

Designe-se por p_1, p_2, p_3, p_4 o número de componentes violadas sem adjacências múltiplas e o número de sub-grafos contraídos com adjacências múltiplas a 2, 3 e 4 componentes violadas respectivamente. O número de terminais em G'' é $p = p_1 + p_2 + p_3 + p_4$. O número de componentes violadas é $|\mathcal{V}| = p_1 + 2p_2 + 3p_3 + 4p_4$.

O número total de adjacências entre F' e as componentes violadas é igual ao número de adjacências entre F'' e T' mais o número de adjacências em cada um dos sub grafos contraídos, ou seja, no pior caso $2(p-1)+4p_2+5p_3+8p_4$. Temos que subtrair a este número os que são duplamente contados nos sub-grafos contraídos: $2p_2 + p_3 + 2p_4$ (os nós cinzentos com setas na figura 4.3).

Tem-se:

$$\sum_{S \in \mathcal{V}} |F \cap \Gamma(S)| \leq 2p_1 + 4p_2 + 6p_3 + 8p_4 - 2 = 2\left(|\mathcal{V}| - 1\right) \leq 2\,|\mathcal{V}|$$

o que conclui a prova do teorema. □

O Teorema 4.1.6 não se aplica para $\tau > 1$, dado que F depende dos conjuntos V^k. Suponha-se que temos uma solução parcial não admissível X em que o número de componentes violadas é p para $k = 1, \cdots, \tau$ e todos os conjuntos $\Gamma^k(S)$ contêm nós que pertencem a todos os V^k. Suponha-se ainda que todas as componentes violadas contêm terminais de todos os tipos e que os conjuntos V^k são tais que qualquer aumento mínimo admissível F de X tem τ caminhos entre quaisquer duas componentes violadas. Então para $k = 1, \cdots, \tau$ a soma na equação (4.7) seria $2\tau(p-1)$ e portanto o Teorema 4.1.1 não garante um factor de aproximação constante para $\tau > 1$.

4.2 Heurística tipo a tipo

Dada uma permutação $i_1, i_2, \ldots, i_\tau$ de inteiros dos tipos $1, 2, \ldots \tau$, a heurística *Tipo a Tipo*, adiante designada por TaT, calcula na iteração k uma solução de

Steiner para o grafo $< V^{i_k} >$, sub-grafo de G induzido por V^{i_k}, actualizando os custos dos nós v da solução encontrada fazendo $w_v = 0$, passando em seguida à próxima iteração. A solução final é obtida tornando minimal (em relação à inclusão) a união dos nós das soluções de Steiner obtidas em cada iteração.

O procedimento pode ser repetido para diferentes permutações dos inteiros $1, 2, \cdots, \tau$, sendo a solução a melhor das obtidas (ver figura 4.4).

Figura 4.4: Heurística TaT

1. $Sol \leftarrow \emptyset$
2. $w(Sol) \leftarrow \infty$
3. $\mathcal{P} \leftarrow$ subconjunto de permutações de $\{1, \cdots, \tau\}$.
4. For all $P \in \mathcal{P}$
   (a) $X \leftarrow \emptyset$
   (b) For all $k \in P$
      i. Construir o grafo $< V^k >$ com custos: 0 se $v \in X$, w_v caso contrário.
      ii. $X^k \leftarrow$ solução de Steiner respeitante a $< V^k >$ e T^k
      iii. $X \leftarrow X \cup X^k$
   (c) Tornar X minimal. Para cada $v \in X \setminus \bigcup_k T^k$ (ordenados aleatoriamente) remover v de X se $X \setminus \{v\}$ for $PLMT$ admissível.
   (d) $w(X) \leftarrow \sum_{v \in X} w_v$
   (e) Se $w(X) < w(Sol)$ então
      i. $Sol \leftarrow X$
      ii. $w(Sol) \leftarrow w(X)$
5. Devolver Sol.

Em cada iteração, o trabalho computacionalmente mais pesado consiste na resolução do problema da árvore mínima de Steiner com pesos nos nós. Este problema pode ser resolvido com recurso ao algoritmo de Klein and Ravi [1995], que dá bons resultados para este tipo de problemas. O algoritmo de Klein e Ravi, baseado no de Rayward-Smith [1983], envolve o cálculo dos

caminhos de custo mínimo entre todos os pares de nós, o que é impraticável quer em termos de quantidade de memória necessária quer em termos de tempo de execução para grafos de grandes dimensões.

Dado que a motivação deste trabalho foi a resolução do PLMT para instâncias de grande dimensão, nos resultados apresentados no Capítulo 5 usamos um algoritmo mais simples e mais rápido para obter soluções admissíveis para o problema da árvore de Steiner com pesos nos nós.

Para calcular a árvore de Steiner com pesos nos nós utilizou-se uma adaptação para os custos nos nós da bem conhecida heurística de Kou et al. [1981] para o problema com custos nas arestas. Dado um grafo G com custos nos nós w e conjunto de terminais T defina-se a rede das distâncias $D(T)$ como o grafo completo cujos nós são os elementos de T e o custo de cada aresta $[u, v]$ é o custo do caminho de custo mínimo que liga o terminal u ao terminal v em G. Note-se que o cálculo do caminho de custo mínimo, num grafo não orientado com custos nos nós, entre os terminais u e v com $w_u = w_v = 0$, se reduz ao cálculo do caminho mais curto entre u e v no grafo orientado obtido pela substituição de cada aresta de G por dois arcos de direcções opostas (i, j) e (j, i) com custos w_j e w_i respectivamente.

A solução, X, é obtida calculando a árvore de suporte de custo mínimo de $D(T)$ e fazendo X igual ao conjunto de nós dos caminhos mais curtos correspondentes às arestas da árvore obtida.

No passo final consideram-se todos os nós de X por ordem aleatória e remove-se o nó i de X se todos os nós de T pertencerem à mesma componente conexa do sub-grafo de G induzido por $X \setminus \{i\}$.

Para uma permutação dos tipos o algoritmo resolve τ problemas de Steiner com custos nos nós. A resolução para cada tipo implica o cálculo do caminho mais curto a partir de $p_k = \left|T^k\right|$ origens para todos os outros terminais no grafo $< V^k >$. O algoritmo de Dijkstra pode ser executado em tempo $O\left(m + n \log n\right)$ onde m é o número de arcos e n o número de nós. Fazendo $\beta_k = 2\left|E^k\right|$, $\alpha^k = \left|V^k\right|$, a complexidade desta tarefa é $O\left(p_k\left(\beta_k + \alpha_k \log \alpha_k\right)\right)$. O cálculo da árvore de suporte mínima é executado sobre um grafo completo com p_k nós e tem complexidade, usando o algoritmo de Prim, $O\left(|E| \log |V|\right)$, ou seja $O\left(p_k^2 \log p_k\right)$. Como $p_k \leq \alpha_k \leq \beta_k$ o tempo de cálculo da solução é claramente dominado pelo cálculo dos caminhos mais curtos. Considerando os τ tipos a complexidade do cálculo da solução é $O\left(\sum_{k=1}^{\tau}\left(p_k\left(\beta_k + \alpha_k \log \alpha_k\right)\right)\right)$ para cada permutação dos tipos.

Tornar a solução X minimal envolve o cálculo das componentes conexas nos grafos induzidos por $X \cap V^k$, sendo necessário recalcula-las para cada

elemento da solução. O tempo necessário para calcular componentes conexas tem complexidade $O(|E| + |V|)$. Não sendo possível determinar à partida o número de nós e arestas dos sub-grafos $\langle X \cap V^k \rangle$, nota-se que no pior caso (i.e. uma solução composta por todos os nós de G) a complexidade seria $O(|V|^2 + |E||V|)$. A experiência mostrou que em geral esta fase do algoritmo é mais rápida do que a da obtenção da solução. O algoritmo pode ser modificado, executando-se esta fase apenas para a melhor solução encontrada entre as permutações consideradas.

4.3 Heurística tipo GRASP

4.3.1 Heurísticas GRASP

As heurísticas do tipo GRASP (Greedy randomized adaptive search procedure) foram introduzidas por Feo and Resende [1989] para o problema dos conjuntos de cobertura e posteriormente formalmente descritas pelos mesmos autores [Feo and Resende, 1995].

Estas heurísticas são algoritmos iterativos em que cada iteração consiste em duas fases: na primeira é construída um solução e na segunda é usado um procedimento de pesquisa local para melhorar a solução. Se a solução da iteração corrente é melhor que as anteriores então é mantida. O processo termina depois de executadas um número de iterações pré-fixado.

Tipicamente na primeira fase é criada uma lista de candidatos a entrar na solução e escolhido um deles aleatoriamente. A fase de pesquisa local melhora a solução obtida de acordo com um critério adequado ao problema (por exemplo eliminando arestas redundantes num problema de conexidade).

O algoritmo pode ser comparado a um processo de amostragem em que em cada iteração se obtém uma amostra da solução. Como em qualquer processo de amostragem quanto maior o número de amostras (iterações) mais provável é a obtenção de um bom resultado (solução).

4.3.2 Heurística GRASP para o PLMT

As heurísticas TaT e PD obtêm uma solução admissível adicionando em cada iteração nós a uma solução corrente não admissível, X. A heurística TaT adiciona a X um conjunto de nós que garantem a ligação de todos os

terminais de um tipo previamente especificado e atribui custo zero a todos os nós já adicionados, uma vez que já estando na solução não há custo adicional por serem reutilizados. A heurística PD adiciona a X, em cada iteração, um único nó pertencente a V^k e adjacente a X^k, para um qualquer k não fixado à partida (escolhido pelo algoritmo). Apresenta-se em seguida uma heurística do tipo GRASP para o PLMT que é de certo modo uma combinação das duas anteriores.

O algoritmo começa com um conjunto X que contém inicialmente todos os terminais de T^k, $k = 1, \cdots, \tau$ e em cada iteração acrescenta nós à solução não admissível X da forma seguinte: selecciona-se aleatoriamente entre os tipos k para os quais existem terminais de T^k não ligados em $< X^k >$, um deles com distribuição uniforme discreta. Em seguida escolhe-se aleatoriamente uma componente conexa S de $< X^k >$, sub-grafo induzido por X^k que inclua terminais do tipo k, e determina-se o caminho de custo mínimo P, entre os caminhos de custo mínimo que ligam S com todas as outras componentes de $< X^K >$ que incluam terminais do tipo k. Os nós de P são adicionados a X e os pesos dos nós são alterados fazendo $w_v = 0$ para todos os nós $v \in P$. Note-se que, dado que os pesos dos nós em X são todos iguais a zero, P pode ser obtido facilmente com o algoritmo de Dijkstra [Dijkstra, 1959], escolhendo arbitrariamente um nó de S como o nó de origem e parando assim que um nó de uma componente de $< X^k >$, que inclua terminais de T^k e seja diferente de S, seja incluído no caminho (ver figura 4.5).

O processo iterativo termina assim que X for solução admissível. O passo seguinte consiste em tornar minimal a solução obtida no processo descrito.

Dado o seu carácter aleatório, repetindo o GRASP um número pré-determinado de vezes produzir-se-ão soluções diferentes. A solução final será a melhor entre as que foram obtidas.

Tal como no caso anterior não é possível analisar a complexidade do tempo de execução do algoritmo. Nota-se apenas que, para além do cálculo das componentes conexas, é também necessário calcular caminhos de custo mínimo. Isto não implica, no entanto, que o tempo de execução desta heurística por iteração, seja necessariamente maior do que o da anterior.

Figura 4.5: Heurística GRASP

1. $w\left(Sol\right) \leftarrow \infty$
2. $r \leftarrow$ número de repetições
3. Para $i = 1, \cdots, r$
   (a) $X \leftarrow \bigcup_k T^k$
   (b) Enquanto X não for $PLMT$ admissível
      i. $X^k = X \cap V^k$. Calcular $\mathcal{C}(\langle X^k \rangle),\ \ k = 1, \cdots, \tau$
      ii. $Q = \{k :$ os terminais de T^k não pertencem todos à mesma $S \in \mathcal{C}(\langle X^k \rangle),\ k = 1, \cdots, \tau\}$
      iii. Se $Q = \emptyset$ terminar o ciclo
      iv. $p \leftarrow$ membro de Q seleccionado aleatoriamente.
      v. $S \leftarrow$ membro de $\mathcal{C}(\langle X^p \rangle) : S \cap T^p \neq \emptyset$ seleccionado aleatoriamente
      vi. $P \leftarrow$ caminho de custo mínimo ligando S to $U \in \mathcal{C}(\langle X^p \rangle) \setminus S : U \cap T^p \neq \emptyset$
      vii. $X \leftarrow X \cup P,\ \ w_v = 0,\ \forall v \in P$
   (c) Tornar X minimal. Para cada $v \in X \setminus \bigcup_k T^k$ (ordenado aleatoriamente) remover v de X se $X \setminus \{v\}$ for $PLMT$ admissível.
   (d) $w(X) \leftarrow \sum_{v \in X} w_v$
   (e) Se $w(X) < w(Sol)$ então
      i. $Sol \leftarrow X$
      ii. $w(Sol) \leftarrow w(X)$
4. Sevolver Sol.

Capítulo 5

Experiências Computacionais

Realizaram-se testes em computador para avaliar a qualidade das soluções produzidas pelas heurísticas, bem como a possibilidade de utilização da formulação baseada em fluxos (3.2a)-(3.2h) em instâncias de pequena dimensão.

5.1 Caso geral

Apresentam-se aqui os dados utilizados para o caso em que os conjuntos V^k não são todos iguais, i.e., o problema não se reduz ao da floresta de Steiner. Foram usados dados reais e dados simulados para testar as heurísticas.

5.1.1 Dados reais

Os dados reais respeitam à ligação de áreas protegidas (AP) da Península Ibérica (PI) com características climáticas similares. PI é representada por 580,696 células de 1km $\times$ 1km, das quais 80,871 pertencem a 681 AP e foram definidas como terminais. Estes foram agrupados em quatro tipos com características climáticas semelhantes, de acordo com quatro variáveis climáticas consideradas fundamentais na distribuição das espécies. Duas células (quadradas) foram consideradas adjacentes se tiverem um lado ou vértice comum, obtendo-se um grafo em grelha cruzada em que cada nó tem oito arestas de ligação. Um grafo deste tipo com dimensões $m \times n$ tem $2(m-1)(n-1) + m(n-1) + n(m-1)$ arestas.

As células com intervenção humana considerável (valores da pegada hu-

mana, disponíveis em http://www.ciesin.columbia.edu/wild_areas, maiores que 60 numa escala de 0 a 100) foram excluídas por se considerar que não são adequadas para movimentação de espécies animais selvagens. Isto reduziu o número de células para 438,948, ficando todas a AP incluídas.

A Figura 5.1 mostra a localização das áreas protegidas para cada uma das quatro classes e das células excluídas devido a valores elevados da pegada humana.

Para $k = 1, \ldots, 4$, V^k foi definido como o conjunto de células que não difere significativamente das condições climáticas médias das AP de classe k. O processo seguido foi o seguinte. Calculou-se o centroide, no espaço climático, das células de cada uma das classes climáticas definidas bem como a distância Euclidiana das condições climáticas de cada célula ao centroide de cada uma das classes. Obtiveram-se valores $d^k(v)$, para cada célula v, expressando a dissemelhança da célula v a cada uma das classes climáticas k. $v \in V^k$ (i.e., v não é uma k-barreira) se $d^k(v)$ é inferior a um determinado valor limite B^k.

Foram considerados dois cenários. No cenário 1, B^k foi definido como a maior dissemelhança $d^k(v)$, entre as células v de cada área protegida de classe k. No cenário 2, B^k representa o terceiro quartil dos valores $d^k(v)$ para as células da áreas protegidas de classe k. A célula u foi incluída em V^k (i.e., u não é uma k-barreira) se u pertence a alguma AP de classe k, ou $d^k(u) < B^k$.

A identificação de ligações entre áreas protegidas climaticamente semelhantes, livres de barreiras climáticas, sustenta-se na hipótese defendida por Alagador et al. [2012] de que espécies com requisitos ecológicos semelhantes ocupam os mesmos ambientes. Assim, ligar áreas protegidas, climaticamente semelhantes, é uma forma efectiva de promover a dispersão das espécies contrariando, em parte, o efeito negativo da fragmentação.

Foi atribuído um custo a cada célula, que não pertença a uma área protegida, proporcional à área desta não coberta pela rede Natura 2000 ($w_v = (100 - \text{percentagem coberta pela rede Natura 2000 de } v)/100$). A rede Natura 2000 é um programa de conservação desenhado para complementar os programas nacionais de áreas protegidas. A todas as células pertencentes a áreas protegidas foi atribuído custo zero.

Detalhes sobre estes dados estão disponíveis em Alagador et al. [2012].

Os problemas resultantes dos cenários 1 e 2 descritos, são designados a partir daqui por $IP1$ e $IP2$ respectivamente.

5.1.2 Dados simulados

Foram feitos dois tipos de simulações. Um utilizando grafos com a mesma estrutura dos dados reais para a Península Ibérica e outro utilizando grafos sem estrutura pré-determinada.

1) Grafos em grelha cruzada. Os dados simulados foram gerados como segue. Cada nó de V é uma célula de uma grelha $n \times n$. Duas células são adjacentes se tiverem um lado ou vértice comum.

Para gerar V^k começa-se por gerar aleatoriamente, com distribuição uniforme discreta, um inteiro $s \in [0, \tau]$ e assume-se que as espécies $1, \ldots, s$ são "especialistas" (só podem sobreviver em condições ambientais muito específicas) e as espécies $s+1, \ldots, \tau$ são "generalistas"(capazes de sobreviver numa gama de condições climatéricas mais vasta). Cada nó v de V é incluído em V_k com probabilidade $1/4$ para cada uma das espécies especialistas $k \leq s$, e com probabilidade $3/4$ para as espécies "generalistas" $k > s$. O número de terminais para cada tipo foi gerado aleatoriamente, com distribuição uniforme discreta, no intervalo $[2, \max\{|V|/1000, 5\}]$. Os terminais foram escolhidos aleatoriamente entre os nós de V^k.

A cada nó $v \in V \backslash (\cup_k T^k)$ foi atribuído um custo aleatório com distribuição $\mathcal{U}[0, 1]$. Aos nós terminais de T^k foi atribuído custo zero.

Foi gerado um conjunto de instâncias pequenas com $n = 10, 20, 30, 40, 50$ e $\tau = 2, 3, 4, 6, 8, 10$ e um conjunto de instâncias grandes com $n = 100, 200, 300, 400, 500$ e $\tau = 4, 6, 8, 10$.

Para cada par de valores n e τ foram geradas 10 instâncias, obtendo-se um total de 500 problemas de teste.

2) Grafos genéricos. Consideraram-se dois métodos para gerar grafos aleatórios. Dado um grafo não orientado com n nós o método de Erdős and Rényi [1959], $G(n, M)$, escolhe aleatoriamente, com distribuição uniforme, M arestas entre as $\frac{n(n-1)}{2}$ possíveis. O método de Gilbert [1959], $G(n, p)$, considera todas as arestas possíveis e inclui-as no grafo com probabilidade p. O número de arestas escolhidas tem distribuição $B(\frac{n(n-1)}{2}, p)$ e portanto o seu valor esperado é $p\frac{n(n-1)}{2}$. Fazendo $p = \frac{M}{n(n-1)/2}$ os dois métodos têm comportamento semelhante à medida que n cresce. Dado que é mais simples de programar optou-se pelo método $G(n, p)$.

Pretende-se obter grafos $G(V, E)$ conexos. Erdős and Rényi [1960] mostraram que para $M \sim c\, n \log(n)$ com $c \leq \frac{1}{2}$ o grafo $G(n, M)$ é *quase certamente* conexo. Fazendo $p = \frac{M}{n(n-1)/2}$ e $c = \frac{1}{2}$ tem-se a expressão equivalente $p \sim \frac{\log n}{n-1}$.

A distribuição assimptótica do grau dos nós de $G(n, p)$ é $Po(np)$, para $n \geq 100$ e $np \leq 10$ [Bettstetter et al., 2010].

O processo de geração dos grafos com n nós e τ tipos foi o seguinte:

1. $p = \frac{\log n}{n-1}$
2. Gerar o grafo pelo método $G(n, p)$. Para todas as arestas possíveis gerar um número aleatório x com distribuição $\mathcal{U}(0, 1)$. Se $x \leq p$ inserir a aresta no grafo.
3. Se o grafo não for conexo repetir o processo.
4. Gerar os conjuntos $V^k, k = 1, \ldots, \tau$. Fixar p_2, probabilidade de $v \in V^k$. Para todos os nós v gerar um número aleatório x com distribuição $\mathcal{U}(0, 1)$. Se $x \leq p_2$ então $v \in V^k$.
5. Gerar os conjuntos de terminais. Para cada tipo k o número de terminais, p_k, foi obtido através de uma distribuição uniforme discreta no intervalo $[2, \max\{n/1000, 5\}]$. Os nós terminais foram obtidos através da escolha aleatória de p_k elementos de cada um dos conjuntos V^k

Geraram-se grafos de pequena dimensão e de grande dimensão, com 5 problemas de cada tipo. Para os grafos pequenos escolheu-se $n = 100, 400, 900, 1600, 2500$, $\tau = 2, 4, 6, 8, 10$ e $p_2 = 0.25, 0.50, 0.75$, perfazendo um total de 375 problemas. Para os de grande dimensão usou-se $n = 10000, 40000, 90000, 160000, 250000$, $\tau = 4, 6, 8, 10$ e $p_2 = 0.25, 0.50, 0.75$, num total de 300 problemas.

5.1.3 Resultados

Apresentam-se em seguida os principais resultados dos testes computacionais efectuados. Todos os tempos nas tabelas referem-se a tempo de CPU.

As heurísticas foram programadas em C++, utilizando a Boost Graph Library para armazenar os grafos e calcular árvores de suporte mínimas, caminhos mais curtos e componentes conexas. Os números aleatórios foram

gerados com recurso à GNU Scientific Library, usando como gerador base o Mersenne Twister. Não foi utilizada programação paralela e portanto foram executadas num único processador. As soluções para as instâncias da Península Ibérica foram obtidas numa máquina com processador Intel Core2 Quad CPU Q9450 @2.66GHz e 4GB de memória, enquanto que para os dados simulados foram obtidas numa máquina com 2 processadores AMD Opteron 6172 processors (24 cores) @2.1GHz e 64GB de memória.

1) Dados reais. A Tabela 5.1 apresenta os resultados obtidos para os dados da Península Ibérica, em cada um dos dois cenários descritos, para cada uma das três heurísticas. A primeira coluna identifica a instância do problema (cenários 1 e 2). Cada um dos três pares de colunas restantes contém o valor da solução obtida com uma heurística: GRASP, tipo a tipo (TaT) e primal-dual (PD), e os tempos de execução, em segundos, correspondentes. A heurística TaT executou todas as permutações dos $\tau = 4$ tipos, enquanto que a GRASP foi limitada a duas horas de execução. Permitiu-se que os programas terminassem a iteração corrente, se tivesse sido iniciada antes do limite de tempo expirar, pelo que os tempos de execução podem exceder os 7200 segundos. A heurística PD não foi limitada por tempo, para que fosse possível obter uma solução.

Tabela 5.1: Resultados para a Península Ibérica.

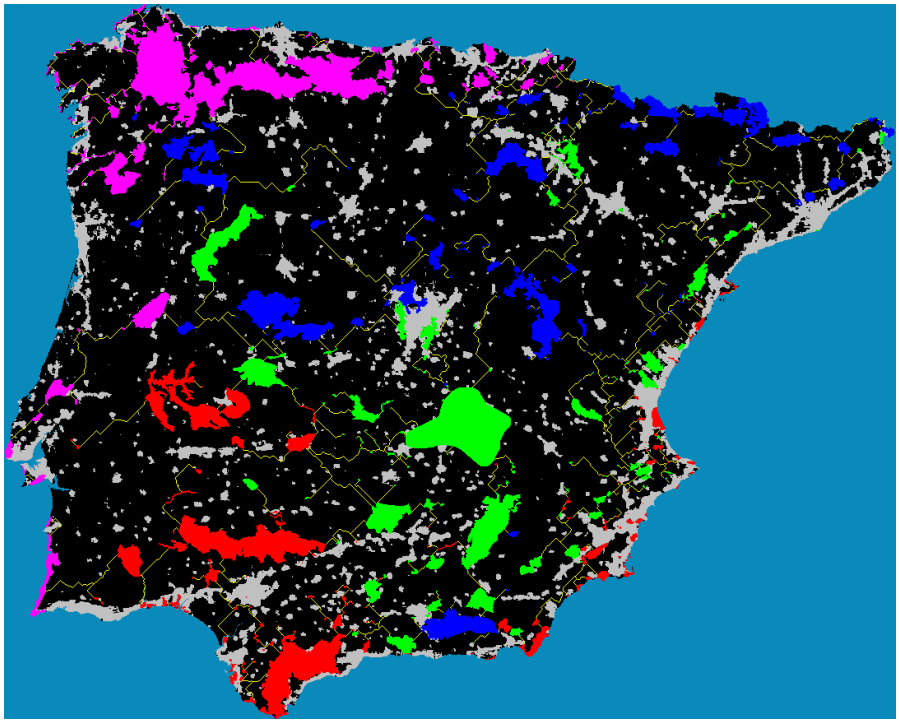
Instância	GRASP		TaT		PD	
	custo	tempo	custo	tempo	custo	tempo
$IP1$	2024.67	7782.6	2035.73	5012.4	2162.11	544490.0
$IP2$	2121.03	7525.3	2148.49	7075.9	2167.62	347003.0

A heurística GRASP obteve as melhores soluções. O custo das soluções obtidas com a TaT foram ligeiramente maiores, mas o tempo para executar as 24 permutações dos quatro tipos foram inferiores às duas horas de execução a que a GRASP foi limitada. PD teve um desempenho fraco. Para obter soluções foram necessários tempos de execução muito elevados e o custo das soluções é superior ao obtido com as outras heurísticas. O mau comportamento da heurística PD pode ser explicado pela estrutura específica destes grafos. Nós muito distantes na grelha são ligados por caminhos muito longos, forçando a heurística a incluir na solução uma grande quantidade de nós até obter uma solução admissível. Obter soluções minimais a partir de um número muito elevado de nós, demora muito tempo e produz soluções de fraca qualidade. As heurísticas GRASP e TaT juntam, em cada iteração,

à solução que está a ser construída os nós de um caminho de custo mínimo ligando dois terminais, o que não introduz muitos nós redundantes.

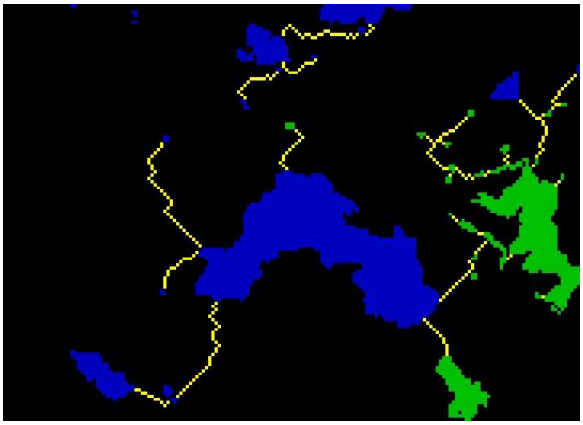
A Figura 5.1 mostra a solução obtida para a instância $IP2$. As áreas protegidas estão coloridas a vermelho, verde, azul e magenta. As manchas cinzentas representam as células que não pertencem a V (pegada humana superior a 60).

Figura 5.1: Dados para a Península Ibérica(cenário 2). Áreas protegidas estão coloridas a vermelho, verde, azul e magenta. k-barreiras são as áreas cinzentas. As células da solução obtida com a heurística TaT estão coloridas a amarelo.

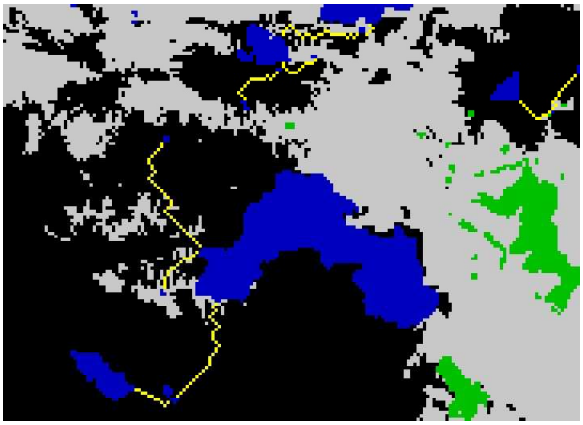


A Figura 5.2 mostra a solução para uma zona do nordeste da Península Ibérica com uma área de $100 \times 100 Km$. Em 5.2a apresenta-se a solução obtida enquanto em 5.2b e 5.2c se apresentam as soluções obtidas para os tipos azul e verde respectivamente. As manchas cinzentas correspondem às k-barreiras para cada um dos tipos. Note-se que o grafo induzido pelo tipo azul não é conexo. As heurísticas obtêm a solução em cada uma das componentes conexas para o tipo respectivo.

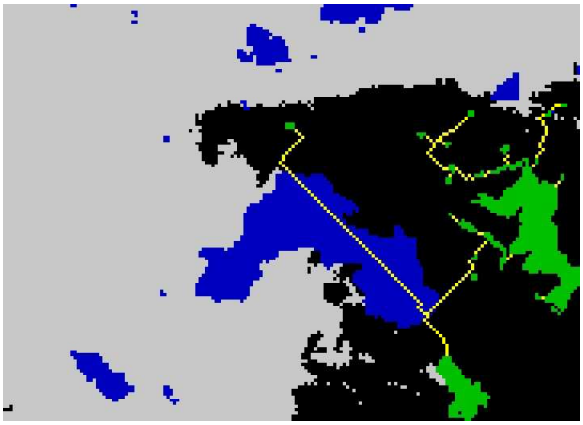
Figura 5.2: Solução para uma área de $100km \times 100km$ do nordeste da Península Ibérica



(a) Solução para os tipos azul e verde



(b) Solução para o tipo azul. Áreas cinzentas são barreiras para o tipo azul.



(c) Solução para o tipo verde. Áreas cinzentas são barreiras para o tipo verde.

2) Dados simulados: grafos em grelha cruzada Os principais resultados obtidos para as instâncias pequenas e grandes, com os dados simulados para os grafos em grelha cruzada, são apresentados nas Tabelas 5.2 e 5.3, respectivamente. Recorda-se que foram criadas 10 instâncias para cada par de valores $|V|$ e τ. Cada linha das Tabelas 5.2 e 5.3 apresenta os resultados médios obtidos com as 10 instâncias.

Foram estabelecidos limites de tempo comuns para as heurísticas: 1 minuto para as instâncias de pequena dimensão e 30 minutos para as de grande dimensão, permitindo-se no entanto que as GRASP e TaT terminassem a última iteração.

Para a maior parte das instâncias pequenas foi possível obter soluções óptimas, a partir da formulação com fluxos multi-produto (3.2a)-(3.2h), com o CPLEX 12.4, utilizando 24 processos paralelos e com o parâmetro "parallel mode" em modo oportunista. Para cada instância limitou-se o tempo de execução do CPLEX a uma hora de relógio, correspondendo a aproximadamente 24 horas de CPU dado que a máquina possui 24 "cores" e foi usada exclusivamente para este fim.

Na Tabela 5.2 a coluna $\#Opts$ indica o número de instâncias para as quais foi possível obter a solução óptima. As duas colunas *% dev. de* indicam, para cada heurística, a média dos desvios relativos (em percentagem) para o valor óptimo (opt) e para o melhor valor obtido com as heurísticas ($melhH$). Os desvios relativos foram obtidos através da expressão $100(h - w^*)/w^*$, onde h é o valor da solução heurística, e w^* é o valor óptimo (opt), ou o mínimo dos valores obtidos com as três heurísticas ($melhH$), respectivamente. O número de valores óptimos relativamente aos quais as médias foram calculadas é apresentado na coluna $\#Opts$. As colunas *melh.* apresentam o número de vezes em que a heurística obteve o melhor resultado, entre os obtidos pelas três heurísticas para a mesma instância. As colunas *tempo* contêm o tempo médio de execução (em segundos) para as heurísticas TaT e PD. Para a GRASP não são apresentados tempos de execução dado que nunca termina antes do tempo especificado.

A Tabela 5.3 é semelhante com a excepção de não existirem colunas respeitantes a comparações com o valor óptimo, dado que o CPLEX não consegue tratar problemas desta dimensão. Assim, as colunas *% dev.* e *melh.* referem-se apenas a comparações com os melhores resultados das soluções heurísticas.

A heurística GRASP foi claramente a melhor nas instâncias consideradas. enquanto que o desempenho da PD foi fraco.

Tabela 5.2: Grafos em grelha cruzada: resultados para instâncias pequenas.

$\|V\|$	τ	#Opts	GRASP			TaT				PD			
			% dev. de			% dev. de				% dev. de			
			opt	melhH	melh.	opt	melhH	melh.	tempo	opt	melhH	melh.	tempo
100	2	10	1.67	1.67	9	1.67	1.67	9	0.00	2.15	2.15	9	0.00
	3	10	1.66	1.66	9	1.66	1.66	9	0.01	0.00	0.00	10	0.00
	4	10	4.02	1.94	8	3.87	1.79	9	0.06	6.49	4.20	6	0.00
	6	10	2.32	2.17	9	3.31	3.16	8	5.19	1.61	1.48	7	0.00
	8	10	1.21	0.00	10	1.21	0.00	10	44.43	2.06	0.86	5	0.01
	10	10	0.31	0.31	9	1.27	1.27	7	60.51	0.75	0.75	8	0.01
400	2	10	0.34	0.00	10	2.51	2.18	5	0.01	7.53	7.17	5	0.02
	3	10	1.80	0.25	9	4.55	2.90	5	0.01	11.47	9.71	4	0.02
	4	10	1.60	0.00	10	4.48	2.79	5	0.06	9.02	7.27	3	0.03
	6	10	2.08	0.95	8	3.56	2.38	7	1.49	9.06	7.87	1	0.04
	8	10	0.90	0.10	9	3.70	2.87	6	18.21	9.50	8.62	2	0.06
	10	10	2.24	0.00	10	6.33	3.95	3	43.46	11.20	8.69	0	0.15
900	2	10	0.85	0.00	10	2.60	1.72	6	0.01	5.95	4.96	6	0.07
	3	10	0.54	0.00	10	1.82	1.28	8	0.02	8.15	7.50	5	0.09
	4	10	1.34	0.00	10	4.66	3.26	4	0.08	15.41	13.82	3	0.14
	6	8	2.71	0.19	9	6.54	4.55	3	3.57	19.06	15.32	2	0.27
	8	7	2.86	0.00	10	4.70	2.05	2	14.31	13.09	13.72	0	0.37
	10	7	1.82	0.39	8	3.25	2.75	5	42.45	10.26	11.07	2	0.56
1600	2	7	1.49	0.22	9	1.92	1.34	4	0.02	7.93	7.66	3	0.22
	3	5	0.09	0.00	10	0.49	4.01	4	0.05	0.65	8.51	4	0.36
	4	6	0.69	0.20	8	1.17	2.38	5	0.21	5.92	10.07	5	0.53
	6	5	1.52	0.43	8	1.93	1.13	5	4.33	4.39	16.84	3	0.74
	8	5	0.29	0.00	10	1.49	2.80	3	12.44	10.56	11.81	2	0.77
	10	1	0.00	0.06	9	1.47	4.27	1	36.13	3.32	15.09	0	1.96
2500	2	4	0.77	0.00	10	0.77	3.35	5	0.04	0.77	5.62	4	0.75
	3	6	0.00	0.08	9	0.98	1.11	6	0.07	0.98	6.91	4	0.67
	4	6	1.52	0.51	9	0.64	2.39	6	0.24	1.96	6.48	4	0.78
	6	4	5.05	0.15	8	5.57	2.41	3	6.01	4.65	7.39	4	1.71
	8	2	1.24	0.00	10	3.35	3.83	2	17.60	22.30	15.73	2	2.60
	10	0		0.04	8		1.59	6	16.20		16.19	3	2.60

Nas instâncias pequenas a média dos 29 valores das colunas que apresentam os desvios médios face ao óptimo na Tabela 5.2 foi 1.48 para a GRASP, 2.81 para a TaT e 7.11 para a PD. Apenas quatro destes 29 valores excederam 2.5% para a GRASP, enquanto que seis valores excederam 4.5% para a TaT. GRASP foi a melhor heurística em pelo menos 8 em dez instâncias para o mesmo $|V|$ e τ. Considerando todas as 300 instâncias pequenas, GRASP foi a melhor heurística em 275 instâncias, TaT em 161 e PD em 116.

Nas instâncias grandes a superioridade da GRASP foi ainda mais evidente. Obteve o melhor resultado em 198 das 200 instâncias. O desvio relativo médio entre os resultados da heurística TaT e o melhor resultado das heurísticas foi sempre inferior a 5.3%, mas obteve o melhor resultado em apenas 24 instâncias.

Os resultados em dados simulados confirmaram o mau comportamento

da heurística PD com este tipo de grafos. Para $|V| \geq 90000$, só foi possível obter soluções num pequeno número de casos, dentro do limite de tempo de 30 minutos fixado (para $|V| = 40000$ e $\tau = 10$ permitiu-se que este limite fosse ligeiramente ultrapassado para se obter resultados). Estes resultados não foram tidos em consideração, ficando em branco na Tabela 5.3, para não enviesar a análise dos resultados. Em geral, as soluções foram de má qualidade. Esta heurística parece ter dificuldades em lidar com instâncias cujos grafos têm a estrutura considerada aqui. A explicação para este comportamento foi apresentada quando foram analisados os resultados para a Península Ibérica.

Deve notar-se que em várias instâncias com valores elevados de τ ($\tau = 8, 10$), a heurística TaT conseguiu terminar a execução dentro do prazo estabelecido. Este comportamento é justificado pela forma como as instâncias foram geradas. Cada célula da grelha $n \times n$ pertence a V^k com probabilidade $1/4$ para uma espécie "especialista" k e $3/4$ para uma "generalista". Sucede por vezes, para uma espécie especialista k, que as componentes conexas do sub-grafo induzido por V^k contêm no máximo um terminal de T^k. Nesta situação não há necessidade de considerar a espécie k. Como as espécies especialistas foram escolhidas aleatoriamente entre as τ espécies consideradas, o número de espécies efectivo é nalguns casos menor que τ.

Tabela 5.3: Grafos em grelha cruzada: resultados para as instâncias grandes.

$\|V\|$	τ	GRASP		TaT			PD		
		%dev.	melh.	%dev.	melh.	tempo	%dev.	melh.	tempo
10000	4	0.00	10	5.27	1	1.56	18.17	1	43.71
	6	0.00	10	4.55	1	49.93	19.91	1	89.57
	8	0.00	10	3.43	2	214.82	11.90	2	67.37
	10	0.00	10	4.86	0	766.98	21.31	0	148.57
40000	4	0.00	10	3.13	2	52.95	14.74	2	1292.83
	6	0.00	10	3.50	0	732.16	7.55	0	1116.29
	8	0.00	10	3.44	1	982.21	14.08	1	1709.82
	10	0.00	10	3.59	1	1320.94	13.60	1	1998.23
90000	4	0.00	10	2.46	3	185.65			
	6	0.00	10	2.21	2	1056.98			
	8	0.00	10	3.66	0	1654.99			
	10	0.13	9	3.73	1	1823.81			
160000	4	0.00	10	2.24	4	819.54			
	6	0.00	10	2.42	2	1325.13			
	8	0.03	9	2.34	2	1931.62			
	10	0.00	10	2.14	0	1947.12			
250000	4	0.00	10	3.26	0	1378.50			
	6	0.00	10	2.45	0	1821.26			
	8	0.00	10	2.76	0	2053.90			
	10	0.00	10	2.09	2	2297.54			

3) Dados simulados: grafos genéricos Os resultados foram obtidos na mesma máquina utilizada para as instâncias com os grafos em grelha cruzada. Os limites de tempo fixados foram os mesmos: 1 minuto para as instâncias de pequena dimensão e 30 minutos para as de grande dimensão.

Os resultados para as instâncias pequenas encontram-se nas tabelas 5.4, 5.5 e 5.6 para $p_2 = 0.25$, $p_2 = 0.50$ e $p_2 = 0.75$ respectivamente.

No caso em que os conjuntos V^k têm menos elementos, 25% dos nós em média, foi possível obter o óptimo para todas as instâncias.

Os resultados mostram que a heurística GRASP obteve a solução óptima na maioria dos casos, 82 em 125, e o desvio médio para o óptimo nunca ultrapassa 2.5%. A GRASP obteve o melhor resultado das heurísticas em 115 casos.

A heurística TaT obteve a solução óptima em 64 dos 125 casos e um desvio médio para o óptimo inferior a 4% excepto em 3 tipos de instâncias. Obteve a melhor solução heurística em 69 casos.

A heurística PD obteve a solução óptima em 45 casos, sendo a melhor heurística apenas nestes casos. Este comportamento observa-se apenas nas instâncias mais fáceis. Para os valores de τ mais elevados os desvios médios face ao óptimo são em geral grandes quando comparados com os das outras heurísticas.

Os tempos de execução médios para a GRASP não são apresentados por serem sempre de 60 segundos. Para a TaT os tempos aumentam com τ como esperado, aproximando-se dos 60 segundos para $\tau \geq 8$. A heurística PD é muito rápida. Globalmente GRASP exibe o mesmo comportamento que nos grafos em grelha cruzada, sendo claramente a melhor heurística.

Quando o número de elementos dos conjuntos V^k aumenta, para valores médios de 50% e 75% dos nós, torna-se mais difícil obter a solução óptima com o CPLEX: 103 e 88 soluções respectivamente, em especial para os problemas de maior dimensão. Para os casos em que foi possível obter o óptimo nota-se claramente uma diminuição da qualidade das soluções heurísticas obtidas, sobretudo com TaT e PD. A superioridade da GRASP é ainda mais evidente nestes casos.

Os resultados para as instâncias de grande dimensão constam das tabelas 5.7, 5.8 e 5.9 para dimensões médias dos V^k de $25\%, 50\%$ e 75% dos nós respectivamente. Para $p_2 = 0.25$ e $p_2 = 0.50$, GRASP é sempre a melhor heurística e a TaT obtém resultados de desvio médio para a GRASP inferiores a 9.05% no primeiro caso e a 12.53% no segundo caso. A heurística PD obtém

Tabela 5.4: Grafos genéricos: resultados para instâncias pequenas com $p_2 = 0.25$.

$\|V\|$	$\overline{\|E\|}$	τ	#Opts	GRASP			TaT				PD			
				% dev. de			% dev. de				% dev. de			
				opt	melhH	melh.	opt	melhH	melh.	tempo	opt	melhH	melh.	tempo
100	239.8	2	5	0.00	0.00	5	0.87	0.87	4	0.00	0.00	0.00	5	0.00
	237.6	4	5	0.00	0.00	5	0.00	0.00	5	0.01	0.00	0.00	5	0.00
	223.4	6	5	0.00	0.00	5	0.00	0.00	5	0.01	0.00	0.00	5	0.00
	214.4	8	5	0.00	0.00	5	0.00	0.00	5	0.05	0.00	0.00	5	0.00
	220.4	10	5	0.00	0.00	5	1.04	1.04	4	0.37	1.04	1.04	4	0.00
400	1177.8	2	5	0.00	0.00	5	0.78	0.78	4	0.00	0.78	0.78	4	0.00
	1208.8	4	5	0.00	0.00	5	1.35	1.35	4	0.00	1.35	1.35	4	0.00
	1234.8	6	5	0.65	0.46	3	1.39	1.21	4	0.25	5.72	5.52	1	0.00
	1189.2	8	5	0.00	0.00	5	0.00	0.00	5	0.27	0.00	0.00	5	0.00
	1209.8	10	5	0.31	0.31	4	0.00	0.00	5	26.42	2.76	2.76	1	0.01
900	3074.2	2	5	0.00	0.00	5	0.00	0.00	5	0.00	0.00	0.00	5	0.00
	3071.8	4	5	1.07	0.00	5	2.86	1.76	2	0.05	2.71	1.62	2	0.01
	3068.2	6	5	0.49	0.40	4	1.30	1.21	3	0.74	2.55	2.46	1	0.02
	3063.8	8	5	0.64	0.59	4	2.64	2.58	0	10.23	4.07	4.02	1	0.02
	3107.4	10	5	2.35	1.08	1	2.00	0.76	3	36.43	4.19	2.90	1	0.03
1600	5898.0	2	5	0.45	0.00	5	3.01	2.52	2	0.00	3.01	2.52	2	0.01
	5906.2	4	5	0.31	0.00	5	8.03	7.72	0	0.09	9.60	9.29	0	0.02
	5893.8	6	5	1.40	0.00	5	3.86	2.42	1	1.65	8.34	6.85	0	0.03
	5846.2	8	5	0.91	0.38	4	3.40	2.87	1	43.53	7.25	6.71	0	0.05
	5867.4	10	5	1.30	0.00	5	3.42	2.09	0	57.97	5.89	4.53	0	0.10
2500	9748.6	2	5	0.64	0.26	4	8.28	7.91	3	0.01	3.16	2.79	2	0.01
	9760.8	4	5	0.29	0.00	5	1.66	1.35	3	0.15	2.06	1.74	3	0.03
	9739.4	6	5	1.11	0.02	4	3.66	2.53	1	8.32	6.02	4.87	0	0.06
	9749.2	8	5	0.87	0.00	5	3.59	2.69	0	50.45	8.36	7.42	0	0.10
	9791.0	10	5	1.78	0.00	5	5.34	3.49	0	50.45	10.57	8.65	0	0.21

resultados fracos, relativamente às outras heurísticas, mas que melhoram com a dimensão dos problemas.

Para as instâncias geradas com $p_2 = 0.75$ a heurística GRASP não apresenta sempre os melhores resultados, sobretudo para V e τ grandes. Nestes casos o número de repetições efectuadas pelo algoritmo no tempo especificado é pequeno, entre 1 a 3, o que viola os pressupostos dos algoritmos deste tipo. Ainda assim, o desvio médio para a melhor heurística é sempre inferior a 1.15%. PD nunca obtém o melhor resultado mas os desvios para a melhor heurística são menores que nos dois casos anteriores. Isto deve-se, sem dúvida, à dificuldade da GRASP obter resultados nestas instâncias no tempo estipulado.

Tabela 5.5: Grafos genéricos: resultados para instâncias pequenas com $p_2 = 0.50$.

$\|V\|$	$\overline{\|E\|}$	τ	#Opts	GRASP			TaT				PD			
				% dev. de			% dev. de				% dev. de			
				opt	melhH	melh.	opt	melhH	melh.	tempo	opt	melhH	melh.	tempo
100	226.2	2	5	0.00	0.00	5	0.58	0.58	3	0.00	0.58	0.58	3	0.00
	234.4	4	5	0.00	0.00	5	3.27	3.27	3	0.01	4.47	4.47	2	0.00
	242.6	6	5	2.68	0.77	4	1.91	0.00	5	0.44	9.51	7.59	1	0.00
	244.0	8	5	4.10	0.28	4	3.82	0.00	5	26.95	7.12	3.29	1	0.00
	242.4	10	5	0.93	0.42	4	2.02	1.51	3	60.37	1.99	1.48	1	0.01
400	1179.8	2	5	1.47	0.57	4	7.97	6.92	2	0.00	2.96	2.04	1	0.01
	1219.4	4	5	4.20	1.13	3	7.62	4.38	2	0.05	11.38	7.95	2	0.01
	1210.2	6	5	2.87	0.30	3	4.29	1.72	3	2.80	11.47	8.77	0	0.02
	1203.8	8	5	1.53	0.00	5	5.71	4.12	0	46.40	13.06	11.34	0	0.02
	1215.6	10	5	5.26	0.00	5	7.24	1.91	0	60.37	19.77	13.89	0	0.05
900	3063.8	2	5	0.64	0.00	5	3.88	3.22	3	0.01	9.56	8.88	2	0.01
	3054.8	4	5	0.53	0.00	5	4.89	4.32	2	0.11	9.59	9.02	0	0.02
	3095.0	6	5	6.43	0.00	5	12.70	5.86	0	6.51	17.82	10.75	0	0.06
	2994.8	8	5	5.10	0.00	5	9.94	4.58	0	59.79	18.22	12.51	0	0.11
	3074.4	10	4	6.83	0.00	5	9.19	3.55	0	60.36	18.97	11.68	0	0.25
1600	5898.0	2	5	0.07	0.00	5	8.49	8.42	2	0.02	11.16	11.09	2	0.02
	5919.8	4	5	2.36	1.17	4	4.22	3.00	3	0.28	19.80	18.38	0	0.07
	5883.6	6	3	5.88	0.00	5	10.05	3.86	0	16.09	23.23	16.23	0	0.15
	5901.4	8	2	10.35	0.21	3	15.19	6.82	1	60.34	21.60	11.14	1	0.29
	5920.8	10	1	7.26	0.00	5	9.85	4.00	0	60.33	12.96	10.58	0	0.64
2500	9814.8	2	5	1.53	1.10	4	0.79	0.39	3	0.03	2.37	1.95	1	0.02
	9840.8	4	4	1.22	0.00	5	6.63	7.83	0	0.63	22.16	19.08	0	0.10
	9736.2	6	3	0.79	0.00	5	7.63	4.54	0	28.43	22.89	16.85	0	0.27
	9862.8	8	1	2.66	0.04	4	3.24	4.29	1	60.31	13.37	11.16	0	0.65
	9837.2	10	0		0.00	5		5.15	0	60.30		18.00	0	0.89

Tabela 5.6: Grafos genéricos: resultados para instâncias pequenas com $p_2 = 0.75$.

$\|V\|$	$\overline{\|E\|}$	τ	#Opts	GRASP			TaT				PD			
				% dev. de			% dev. de				% dev. de			
				opt	melhH	melh.	opt	melhH	melh.	tempo	opt	melhH	melh.	tempo
100	226.8	2	5	1.07	0.99	4	1.42	1.35	4	0.00	3.84	3.77	3	0.00
	234.4	4	5	2.45	0.00	5	6.56	4.00	3	0.03	8.08	5.48	0	0.00
	240.2	6	5	0.21	0.21	4	0.18	0.18	4	1.04	8.07	8.07	0	0.01
	221.8	8	5	0.00	0.00	5	0.00	0.00	5	60.29	8.78	8.78	2	0.01
	232.8	10	5	0.00	0.00	5	5.26	5.26	2	60.29	1.02	1.02	3	0.01
400	1198.2	2	5	4.21	0.00	5	10.08	5.64	2	0.01	9.90	5.34	2	0.01
	1193.0	4	5	6.87	0.00	5	9.25	2.24	3	0.11	10.99	4.02	0	0.02
	1186.0	6	5	1.22	0.00	5	5.46	4.19	1	5.29	9.29	8.03	0	0.05
	1218.0	8	5	1.12	0.00	5	6.33	5.19	0	60.28	10.67	9.46	0	0.13
	1189.0	10	5	1.59	0.00	5	9.82	8.13	0	60.28	11.21	9.49	0	0.11
900	3067.8	2	5	0.32	0.00	5	7.52	7.21	3	0.01	12.62	12.30	1	0.02
	3028.4	4	5	2.49	0.15	3	7.33	4.87	2	0.31	11.62	9.17	1	0.09
	3071.2	6	4	3.71	0.00	5	6.23	3.31	1	14.92	28.84	21.94	0	0.22
	3040.4	8	4	4.25	0.38	4	8.32	4.25	1	60.28	14.28	8.88	0	0.48
	3079.8	10	2	3.80	0.00	5	8.63	9.12	0	60.27	18.38	11.68	0	0.55
1600	5919.2	2	5	2.11	0.86	3	3.64	2.37	3	0.03	4.74	3.50	2	0.04
	5927.0	4	5	4.00	0.00	5	11.69	7.36	0	0.69	29.59	24.61	0	0.19
	5872.8	6	2	7.06	0.73	4	13.76	3.18	1	31.23	26.20	15.10	0	0.55
	5920.6	8	0		0.00	5		5.54	1	60.27		15.18	0	1.30
	5861.8	10	0		0.00	5		8.45	0	60.26		7.55	0	1.78
2500	9838.8	2	4	2.16	0.00	5	7.51	5.88	1	0.06	7.58	5.82	0	0.05
	9757.0	4	2	2.09	0.00	5	7.93	7.46	0	1.27	26.55	17.72	0	0.45
	9773.6	6	0		0.00	5		5.79	0	58.05		11.10	0	0.80
	9744.2	8	0		0.00	5		7.76	0	60.22		15.54	0	1.29
	9735.6	10	0		0.00	5		9.34	0	60.20		13.36	0	3.81

Tabela 5.7: Grafos genéricos: resultados para as instâncias grandes com $p_2 = 0.25$.

$\|V\|$	$\overline{\|E\|}$	τ	GRASP		TaT			PD		
			%dev.	melh.	%dev.	melh.	tempo	%dev.	melh.	tempo
10000	46183.6	4	0.00	5	7.38	0	2.04	9.99	0	1.22
	46120.6	6	0.00	5	8.50	0	103.27	14.81	0	2.55
	46137	8	0.00	5	8.74	0	1593.90	15.31	0	1.90
	46023.2	10	0.00	5	6.85	0	1800.42	13.31	0	4.29
40000	211608.2	4	0.00	5	9.05	0	26.69	11.32	0	64.42
	211982.8	6	0.00	5	8.93	0	1256.24	13.49	0	129.81
	211897.8	8	0.00	5	7.43	0	1801.49	13.77	0	412.66
	211929.6	10	0.00	5	7.47	0	1802.86	15.58	0	521.18
90000	513118.2	4	0.00	5	9.57	0	157.02	10.72	0	837.69
	513715	6	0.00	5	7.66	0	1804.93	12.36	0	1380.91
	513590	8	0.00	5	7.25	0	1805.61	12.49	0	2802.76
	513667	10	0.00	5	7.53	0	1808.16			
160000	959349.6	4	0.00	5	8.30	0	488.11			
	959622.2	6	0.00	5	8.32	0	1811.64			
	958896.2	8	0.00	5	7.37	0	1821.54			
	958607.2	10	0.00	5	6.86	0	1823.43			
250000	1554280.2	4	0.00	5	7.46	0	1713.03			
	1553107.6	6	0.00	5	7.58	0	1819.03			
	1552357.4	8	0.00	5	7.27	0	1843.25			
	1554449.8	10	0.00	5	6.99	0	1913.61			

Tabela 5.8: Grafos genéricos: resultados para as instâncias grandes com $p_2 = 0.50$.

$\|V\|$	$\overline{\|E\|}$	τ	GRASP		TaT			PD		
			%dev.	melh.	%dev.	melh.	tempo	%dev.	melh.	tempo
10000	46145.0	4	0.00	5	12.53	0	7.41	18.16	0	2.08
	45975.0	6	0.00	5	6.38	0	371.23	21.53	0	6.09
	46173.0	8	0.00	5	7.40	0	1801.16	21.85	0	15.45
	46192.0	10	0.00	5	7.19	0	1801.27	18.46	0	25.80
40000	211937.6	4	0.00	5	10.97	0	106.17	11.41	0	378.84
	212027.2	6	0.00	5	7.83	0	1804.36	14.95	0	670.54
	211898.6	8	0.00	5	6.37	0	1804.13	15.59	0	1361.76
	212080.2	10	0.00	5	6.45	0	1807.59	11.64	0	1962.91
90000	513224.8	4	0.00	5	7.51	0	684.10	11.73	0	2479.40
	513418.2	6	0.00	5	6.83	0	1818.19			
	513700.0	8	0.00	5	4.91	0	1833.43			
	513818.8	10	0.00	5	4.39	0	1849.09			
160000	958301.8	4	0.00	5	7.06	0	1639.29			
	957846.0	6	0.00	5	6.10	0	1865.32			
	958276.8	8	0.00	5	5.15	0	1874.42			
	959033.2	10	0.00	5	3.61	0	1949.45			
250000	1552678.8	4	0.00	5	7.50	0	1889.04			
	1554465.8	6	0.00	5	5.51	0	2019.49			
	1554286.8	8	0.00	5	4.24	0	2050.99			
	1553027.8	10	0.00	5	3.23	0	2178.57			

Tabela 5.9: Grafos genéricos: resultados para as instâncias grandes com $p_2 = 0.75$.

$\|V\|$	$\overline{\|E\|}$	τ	GRASP		TaT			PD		
			%dev.	melh.	%dev.	melh.	tempo	%dev.	melh.	tempo
10000	45952.4	4	0.00	5	7.02	0	21.82	13.86	0	7.19
	46017.4	6	0.00	5	4.53	0	855.65	12.21	0	50.76
	45981.2	8	0.00	5	5.53	0	1801.24	15.42	0	61.07
	45978.6	10	0.00	5	7.81	0	1801.55	11.08	0	82.50
40000	211941.0	4	0.00	5	6.86	0	255.61	9.87	0	467.75
	211823.0	6	0.00	5	4.75	0	1812.34	8.00	0	2314.20
	211464.0	8	0.00	5	3.09	0	1805.78	11.14	0	2975.70
	211761.6	10	0.19	3	1.26	2	1809.00			
90000	513047.2	4	0.00	5	6.24	0	1076.87			
	513422.4	6	0.00	5	1.33	0	1855.32			
	513285.4	8	0.00	5	2.58	0	1856.35			
	513651.8	10	0.10	4	1.25	1	1876.10			
160000	958503.4	4	0.00	5	4.64	0	1877.89			
	959490.0	6	0.00	5	1.30	0	1877.27			
	958935.6	8	0.41	2	0.71	3	2051.18			
	958774.2	10	0.58	3	0.43	2	2125.54			
250000	1553667.0	4	0.00	5	4.18	0	1959.93			
	1553243.8	6	0.05	4	1.87	1	2155.99			
	1554186.8	8	0.78	2	0.71	3	2322.65			
	1553506.0	10	1.15	1	0.10	4	2617.60			

5.2 Efeito da densidade dos grafos genéricos nas heurísticas

A densidade dos grafos em grelha cruzada é, para o caso de uma quadrícula $n \times n$, $4(n-1)(2n-1)/(n^2(n^2-1))$. Para os grafos genéricos utilizados o valor esperado da densidade é $p = \log(n)/(n-1)$. Nos resultados apresentados anteriormente a densidade dos grafos era muito baixa. Referiu-se anteriormente que os maus resultados da heurística PD se deviam ao facto de os terminais se encontrarem "muito longe" uns dos outros, característica inerente a grafos esparsos.

Para avaliar o efeito da densidade dos grafos genéricos no comportamento das heurísticas, geraram-se instâncias com $|V| = 30000, 40000$ e 50000 nós e densidades esperadas de $p = 0.001, 0.002, \cdots, 0.009$. O valor de τ foi fixado em 6 e a probabilidade de um nó pertencer a V^k em $p_2 = 0.5$. Para cada $(|V|, p_1, \tau, p_2)$ foram gerados 5 grafos num total de 135 problemas. Os valores de $|V|$ e p_2 foram escolhidos por forma a que as heurísticas GRASP e TaT pudessem executar um número significativo de iterações. O tempo de execução foi limitado a uma hora.

A Tabela 5.10 apresenta os resultados obtidos. As colunas têm o mesmo significado que anteriormente com excepção das colunas nRep que apresenta o número médio de iterações executadas pela GRASP, nPerm onde se apresenta o número médio de permutações tratadas pela TaT e as três colunas finais em que se apresentam as médias dos rácios h^i/h^j para as três combinações possíveis de heurísticas. Nestas colunas, valores superiores a 1 significam que a heurística h^i teve pior comportamento que a heurística h^j. Os tempos de execução, em segundos, são apresentados apenas para a heurística PD dado que nos outros dois casos foram sempre de uma hora.

A heurística PD é sempre muito rápida, menos de 15 minutos em todos os casos. Comparando a GRASP com a PD verifica-se que esta apresenta sempre melhores resultados que a PD, embora o rácio GRASP/PD se aproxime de 1 à medida que a densidade aumenta (ver Figuras 5.3, 5.4 e 5.5). O comportamento da heurística GRASP relativamente à TaT não apresenta variações significativas, nos grafos considerados, em função da densidade.

Comparando a heurística TaT com a PD observa-se que à medida que a densidade aumenta a segunda obtém melhores resultados. Isto é particularmente evidente para $|V| > 30000$.

Estas observações reforçam a ideia já exposta que a heurística PD não

Tabela 5.10: Grafos genéricos: efeito da densidade.

$\|V\|$	p	GRASP (G)			TaT (T)			PD (P)			G/T	G/P	T/P
		%dev.	melh.	nIter	%dev.	melh.	nPerm	%dev.	melh.	tempo			
30000	0.001	0.00	5	1942	7.87	0	291	11.61	0	203.10	0.93	0.90	0.97
	0.002	0.00	5	1005	6.77	0	152	9.09	0	183.33	0.94	0.92	0.98
	0.003	0.00	5	805	7.63	0	109	9.90	0	156.39	0.93	0.91	0.98
	0.004	0.00	5	647	6.14	0	82	8.84	0	105.87	0.94	0.92	0.98
	0.005	0.00	5	573	5.95	1	66	7.75	1	90.62	0.95	0.93	0.99
	0.006	0.17	4	391	6.57	1	62	7.06	0	95.24	0.94	0.94	1.00
	0.007	0.81	4	381	6.55	0	49	2.01	3	51.59	0.95	0.99	1.05
	0.008	0.00	5	345	6.35	0	43	5.59	0	83.76	0.94	0.95	1.01
	0.009	1.56	3	274	8.29	0	43	3.00	3	53.39	0.94	0.99	1.05
40000	0.001	0.00	5	822	8.81	0	171	14.91	0	292.55	0.92	0.87	0.95
	0.002	0.00	5	485	8.89	0	103	9.49	0	228.62	0.92	0.92	1.00
	0.003	0.00	5	379	5.79	0	64	6.66	0	218.84	0.95	0.94	0.99
	0.004	0.61	4	269	7.80	0	40	4.59	1	243.36	0.93	0.96	1.03
	0.005	0.50	4	213	5.63	1	37	6.18	0	179.70	0.95	0.95	1.00
	0.006	0.00	5	177	5.77	0	30	4.22	1	162.68	0.95	0.96	1.02
	0.007	0.22	4	155	7.21	0	24	5.47	1	147.16	0.94	0.95	1.02
	0.008	0.58	4	154	6.15	1	24	6.36	2	105.74	0.95	0.95	1.00
	0.009	1.25	4	130	5.97	1	20	4.16	0	109.25	0.96	0.97	1.02
50000	0.001	0.00	5	673	8.35	0	80	11.78	0	859.05	0.92	0.90	0.97
	0.002	0.00	5	399	8.93	0	55	7.97	0	473.00	0.92	0.93	1.01
	0.003	0.00	5	266	7.43	0	33	5.70	0	307.22	0.93	0.95	1.02
	0.004	0.00	5	212	5.69	0	22	2.96	0	361.21	0.95	0.97	1.03
	0.005	0.42	4	168	7.96	0	19	6.14	1	254.05	0.93	0.95	1.02
	0.006	0.25	4	132	5.15	0	13	3.44	2	260.04	0.95	0.97	1.02
	0.007	0.78	3	115	5.76	0	13	1.97	3	152.38	0.95	0.99	1.04
	0.008	1.14	4	130	4.75	2	11	5.18	1	211.27	0.97	0.96	1.00
	0.009	0.00	5	111	6.27	0	12	2.88	3	117.65	0.94	0.97	1.03

tem um bom comportamento em grafos que modelam a disposição física no terreno de habitats de espécies. Como o grau dos nós é necessariamente baixo, porque as adjacências se limitam às células de terreno fronteiras, a densidade destes grafos é muito pequena. Em grafos mais densos a heurística comporta-se bem, face às outras duas, sendo uma alternativa viável para obter soluções em problemas de outros tipos.

Figura 5.3: Grafos genéricos: efeito da densidade para $|V| = 30000$

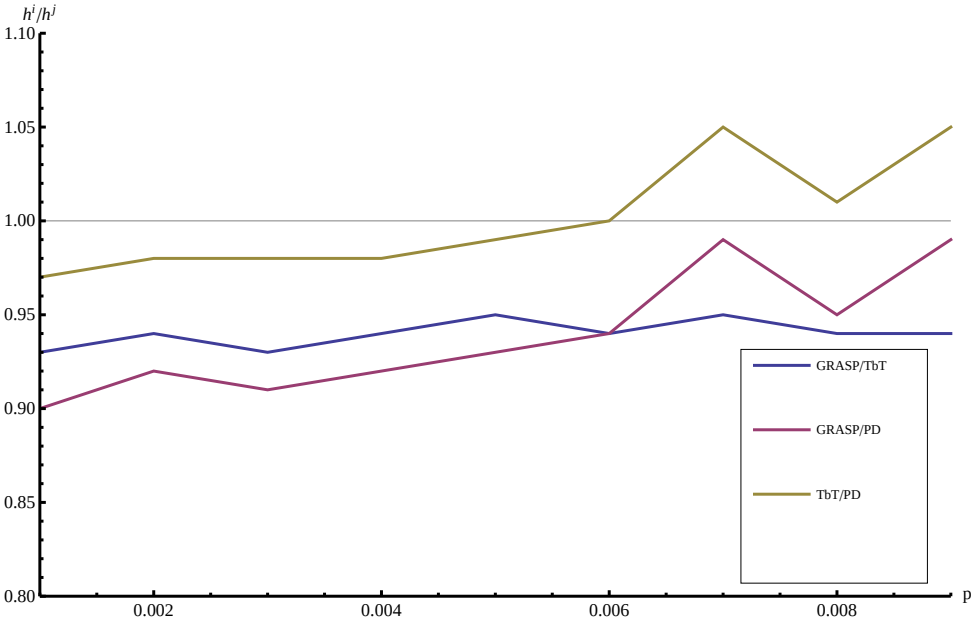


Figura 5.4: Grafos genéricos: efeito da densidade para $|V| = 40000$

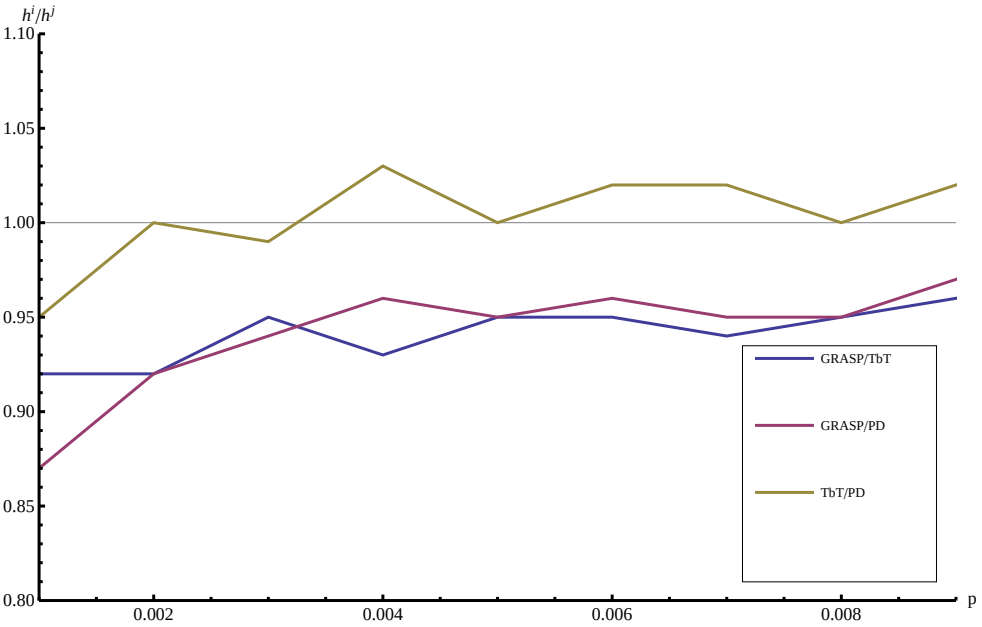
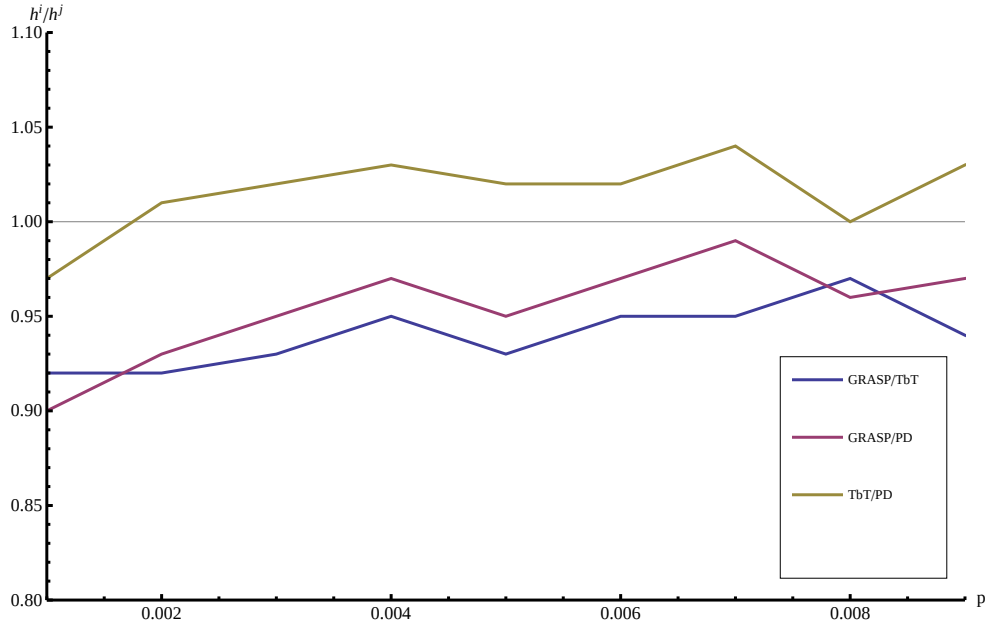


Figura 5.5: Grafos genéricos: efeito da densidade para $|V| = 50000$



5.3 Floresta de Steiner com pesos nos nós

PLMT é uma generalização dos problemas da árvore de Steiner e da floresta de Steiner com pesos nos nós. As heurísticas apresentadas, GRASP, TaT e PD, podem resolver sem qualquer modificação estes problemas.

Para aferir o desempenho das heurísticas na solução do problema da floresta de Steiner com pesos nos nós, caso em que todos os V^k são iguais, efectuaram-se algumas experiências computacionais comparando GRASP e TaT com outras heurísticas para este problema.

Para o problema da floresta de Steiner com pesos nos nós a heurística PD é o algoritmo de Demaine et al. [2009] que provamos ter um factor de aproximação de 2 para o caso dos grafos em grelha cruzada. Outra heurística, baseada no algoritmo de Rayward-Smith[Rayward-Smith, 1983, Rayward-Smith and Clare, 1986], e que dá bons resultados práticos é a de Klein and Ravi [1995].

A heurística de Klein e Ravi (KR) começa por calcular uma matriz M, com os custos dos caminhos de custo mínimo entre todos os pares de nós de V. Inicialmente a solução, X, contém todos os terminais, i.e. todos os nós pertencentes a T^k, $k = 1, \cdots, \tau$. Em cada iteração o algoritmo junta a X os nós de caminhos de custo mínimo entre um nó seleccionado e algumas das componentes conexas de $\langle X \rangle$, o sub-grafo induzido por X. As componentes a ligar são escolhidas recorrendo a uma função f calculada como segue para cada nó $v \in V$. Seja $\mathcal{S}$ o conjunto de componentes conexas de $\langle X \rangle$. Por construção as componentes incluem pelo menos um nó de T^k mas não todos. Seja $\mathcal{S}_r$ a família de todos os r conjuntos de $\mathcal{S}$ (i.e., $|S_r \in \mathcal{S}_r| = r$). Para cada $v \in V$ e $S_r \in \mathcal{S}_r$, seja $w(v, S_r)$ a soma dos custos dos caminhos de custo mínimo que ligam v com cada uma das r componentes em S_r. Para cada $v \in V$, defina-se $f(v, r) = \min_{S_r} w(v, S_r) - (r-1)w_v$. O valor $f(v, r)$ é o custo mínimo de juntar r componentes de $\mathcal{S}$ com r caminhos com origem em v. Note-se que o cálculo de $f(v, r)$ pode ser obtido rapidamente através da matriz M. Finalmente $f(v) = \min_{1 \leq r \leq |\mathcal{S}|} f(v, r)/r$ é o mínimo dos valores médios de $f(v, r)$ com respeito a r. Em cada iteração, KR junta a X os nós dos caminhos com origem em v que minimizam $f(v)$, enquanto $\mathcal{S}$ for não vazio. O último passo consiste em tornar minimal (relativamente à inclusão) a solução X.

Comparou-se o desempenho das heurísticas GRASP, TaT, Demaine et al. [2009] (PD) e Klein and Ravi [1995] (KR) em instâncias de dados simulados, geradas da mesma forma que para os grafos em grelha cruzada, com a ex-

Tabela 5.11: Resultados para a floresta de Steiner.

$\|V\|$	τ	GRASP		TaT		PD		KR	
		custo	tempo	custo	tempo	custo	tempo	custo	tempo
2500	2	19.32	1800.28	20.12	0.11	20.09	1.15	19.20	2.04
		12.61	1800.37	13.01	0.08	15.92	2.34	14.67	1.92
	4	24.45	1800.18	25.31	2.64	26.36	1.57	25.04	2.68
		22.78	1800.48	23.23	2.46	25.53	1.87	22.98	2.33
	6	24.39	1800.65	24.83	104.46	25.38	0.82	24.64	2.73
		30.03	1800.47	30.74	98.04	31.56	1.47	30.44	2.85
	8	36.56	1801.00	38.66	1800.60	38.32	1.67	36.86	4.58
		29.65	1800.23	30.27	1800.37	30.50	0.81	29.97	3.57
	10	34.30	1800.63	37.12	1801.13	36.90	1.82	35.51	3.82
		33.46	1801.23	34.80	1800.23	33.51	1.01	33.30	4.71
10000	2	46.64	1800.54	48.39	0.73	48.02	17.43	46.83	39.35
		42.19	1800.91	43.96	0.78	43.16	25.60	41.97	48.66
	4	62.06	1800.88	65.60	18.54	65.36	26.49	61.77	58.09
		47.44	1801.16	50.20	12.94	49.05	22.79	47.35	48.61
	6	78.49	1800.33	79.95	956.15	81.00	42.66	77.16	64.57
		67.07	1801.34	68.97	655.02	68.02	18.02	67.31	60.66
	8	74.07	1800.78	75.25	1801.68	75.34	17.31	71.36	72.82
		79.73	1800.39	81.12	1801.94	82.17	22.02	78.56	82.25
	10	100.02	1801.10	100.70	1802.77	99.83	59.72	98.07	96.04
		90.77	1801.33	95.66	1802.23	92.31	32.39	89.00	96.32
40000	2	178.80	1801.51	192.27	9.94	189.42	501.73	189.39	1541.14
		83.15	1801.27	87.51	3.47	89.15	328.08	84.28	1154.73
	4	257.27	1806.26	265.71	190.69	253.92	1155.15	273.20	2280.98
		188.07	1800.93	197.61	151.75	187.33	349.34	182.06	1666.43
	6	337.71	1813.55	348.49	1814.00	327.30	882.97	317.66	4884.35
		225.93	1803.42	236.66	1806.29	228.74	829.09	218.27	2187.45
	8	316.43	1815.54	327.89	1821.25	303.15	686.42	292.69	4239.43
		305.71	1802.64	322.56	1812.95	300.84	515.96	291.56	3982.47
	10	297.26	1813.47	310.58	1813.78	290.70	885.06	283.34	3827.51
		372.83	1804.53	386.94	1817.54	360.27	597.45	345.79	6483.80

cepção que $V^k = V$, para $k = 1, \ldots, \tau$. Obtiveram-se grafos em grelha $n \times n$ para $n = 50, 100, 200$ e $\tau = 2, 4, 6, 8, 10$ tipos de terminais. Para cada valor de n e τ foram geradas duas instâncias.

A Tabela 5.11 apresenta os custos e os tempos de execução (em segundos) para cada instância. GRASP e TaT foram limitados a 30 minutos de tempo de execução. O cálculos foram executados na mesma máquina que tratou os casos anteriores de dados simulados.

Os resultados obtidos com as heurísticas GRASP e KR são muito semelhantes. KR obteve a melhor solução em 56.7% dos casos, enquanto GRASP foi a melhor heurística em 40.0% dos casos e PD num caso (3.3%). TaT nunca obteve o melhor resultado. O desvio relativo médio entre a heurística h e a KR, medido por $(h - KR)/KR$, foi de 0.5% para a GRASP, 4.4% para a TaT e 3.5% para a PD.

Considerando apenas os casos em que KR foi melhor que as outras heurís-

ticas ($h > KR$), a média do desvio relativo foi de 3.1% para a GRASP, 5.2% para a TaT e 3.9% para a PD.

Os resultados mostram que a heurística GRASP teve melhor desempenho que KR nas instâncias de menor dimensão, enquanto que, em geral, KR foi melhor nas de maior dimensão. O desvio relativo não excedeu 8.1% (para uma instância com $n = 200$ e $\tau = 8$).

Os valores obtidos com a heurística TaT foram ligeiramente piores do que os obtidos pela GRASP. Isto foi mais evidente para instâncias grandes ($n = 200$).

PD obteve bons resultados nas instâncias de maior dimensão. Para $n = 200$ e $\tau \geq 4$ os resultados são melhores do que para a GRASP, com tempos de execução razoáveis.

A heurística KR mantém em memória a matriz dos custos dos caminhos de custo mínimo entre todos os pares de nós de V. Para $n = 200$ são necessários 6GB de memória enquanto que para $n = 300$ são necessários 30GB. Isto limita a dimensão dos problemas em que KR pode ser utilizada, não sendo possível obter soluções para as instâncias $IP1$ e $IP2$ apresentadas acima.

Dadas as limitações de KR descritas, GRASP e PD são uma boa opção para resolver problemas da floresta de Steiner com pesos nos nós de grande dimensão, para os tipos de grafos considerados.

Capítulo 6

Aplicação informática

Durante a elaboração da dissertação foi desenvolvida uma aplicação informática de código aberto, MultyLink [Brás et al., 2013], disponibilizada para os investigadores interessados através da licença para aplicações livres GPL V3.

A aplicação está disponível em http://purl.oclc.org/multylink

Existem vários métodos que usam a teoria dos grafos para a identificação de áreas economicamente eficientes para promover a conexidade entre os habitats. Alguns destes métodos abordam os problemas de conexidade e representação das espécies simultaneamente. Exemplos são a selecção de áreas para atingir as metas de representação para as espécies, formando uma rede única contígua [Önal and Briers, 2005, Cerdeira et al., 2005, Fuller et al., 2006, Cerdeira et al., 2010], ou permitir formas menos rígidas de coerência espacial, seleccionando áreas agrupadas espacialmente, mas não necessariamente ligadas por corredores contíguos [Önal and Briers, 2002, 2003, Alagador and Cerdeira, 2007]. Outros métodos envolvem apenas a ligação de conjuntos de reservas existentes ou habitats. A Tabela 6.1 apresenta algumas aplicações de código aberto que implementam as abordagens citadas.

Algumas das aplicações referidas na Tabela 6.1 produzem como resultado uma única (ou um pequeno conjunto de) "melhor"ligação (isto é, as áreas que estabelecem a conexidade entre habitats), enquanto outras dão como resultado uma avaliação quantitativa da adequação de cada área para as metas de conexidade estabelecidas.

Nenhum dos métodos acima descritos foi especificamente concebido para ligar eficientemente habitats ocupados por múltiplas espécies, com padrões de distribuição distintos, isto é, que ocorrem em conjuntos distintos de habitats

Tabela 6.1: Aplicações de código aberto para ligação de habitats naturais.

Aplicação	Referência
Circuitscape	McRae and Beier [2007]
	McRae and Shah [2011]
Conefor Sensinode	Saura and Torné [2009]
Connectivity Analysis Toolkit	Carroll et al. [2012]
Linkage Mapper	McRae and Shah [2011]
LQGraph	Fuller and Sarkar [2006]
UNICOR	Landguth et al. [2012]

adequados isolados, e com limitações de dispersão diferentes. Como as áreas adequadas à dispersão de uma espécie poderão ser barreiras à dispersão de outra, as aplicações citadas podem produzir soluções sem qualquer interesse prático.

Embora os métodos existentes para a identificação de ligações possam ser usados para várias espécies (unindo as soluções para cada espécie individual numa única solução), a configuração resultante estaria provavelmente muito longe de uma solução de custo (ou área) mínimo.

MulTyLink é uma aplicação para ligar, de forma eficiente, os habitats ocupados por várias espécies com diferentes distribuições e/ou requisitos de dispersão, e especificamente projectado para lidar com grandes conjuntos de dados. MulTyLink constrói um grafo para cada grupo de espécies "semelhantes", tendo em conta as áreas que actuam como barreiras e a capacidade de dispersão das espécies. O utilizador pode seleccionar a heurística GRASP ou *Tipo a Tipo* para obter uma solução. A aplicação utiliza multi processamento simétrico, sempre que possível, para optimizar o desempenho dos algoritmos.

Os dados necessários para utilizar o MulTyLink são:

- Coordenadas geográficas de cada uma das células de terreno;
- Para cada célula de terreno:
  - Custo;
  - Fricção. Quantifica a resistência da célula à sua utilização, por exemplo a pegada humana;
  - Tipos para os quais a célula é um terminal;

  - Tipos para os quais a célula é uma barreira;

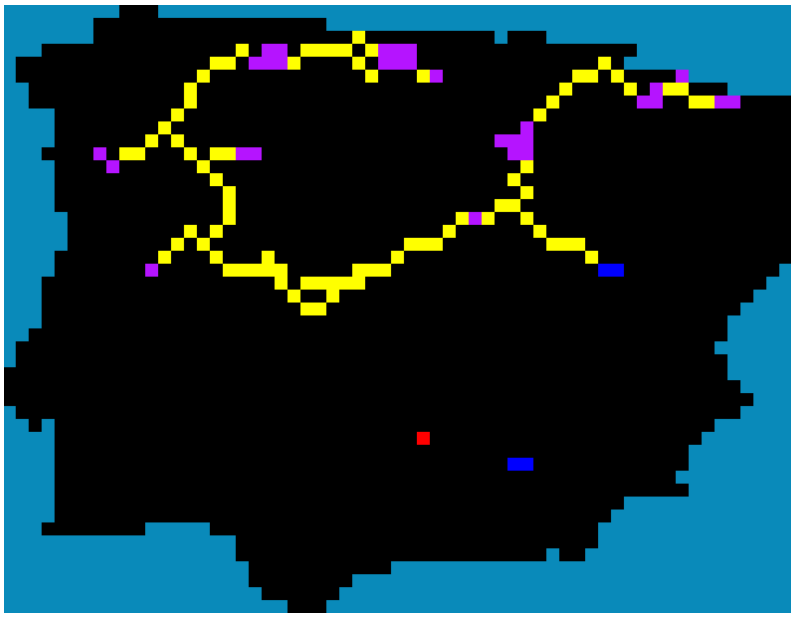
- Distâncias de adjacência (d_k) para cada tipo $k = 1, \cdots, m$. Para um dado tipo, duas células são consideradas adjacentes se a distância euclidiana das suas coordenadas geográficas for inferior ao valor especificado. Com $d_k = 1$ obtém-se um grafo planar em grelha. $d_k = \sqrt{2}$ cria um grafo em grelha cruzada. Valores de $d_k \geq 2$ produzem grafos em que células sem fronteiras comuns são consideradas adjacentes.

As figuras 5.1 e 5.2 foram obtidas com a aplicação desenvolvida, utilizando $d_k = \sqrt{2}$.

A flexibilidade na definição das adjacências das células de terreno, permite obter soluções considerando diferentes estruturas do grafo.

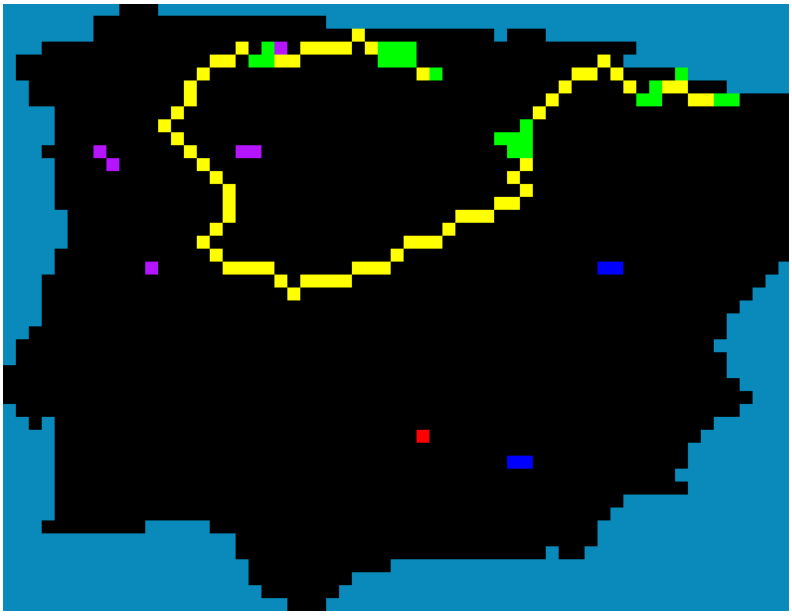
Os exemplos das figuras 6.1 a 6.4 mostram as soluções obtidas para a ligação dos habitats de 3 espécies de animais na Península Ibérica: Lacerta Bilineata a verde, Lacerta Schreiberi a vermelho e Coronella Austriaca a azul. Os habitats com mais de uma espécie estão representados a magenta. A solução é mostrada a amarelo.

Figura 6.1: 3 espécies. Grafo em grelha cruzada



A figura 6.1 apresenta a solução global considerando um grafo em grelha cruzada. Dois dos habitats estão isolados devido ao efeito da pegada humana.

Figura 6.2: Solução para a espécie Lacerta Bilineata



A figura 6.2 apresenta a mesma solução mostrando apenas a espécie Lacerta Bilineata.

Na figura 6.3 mostra-se a solução quando se considera $d_k = 1$, isto é um grafo em grelha. Na figura 6.4 utilizou-se $d_k = 2$, considerando que os animais se podem deslocar saltando uma célula de terreno. Nesta solução um dos habitats deixa de estar isolado. As células de terreno incluídas na solução não são contíguas embora sejam nós adjacentes do grafo.

Figura 6.3: 3 espécies. Grafo em grelha

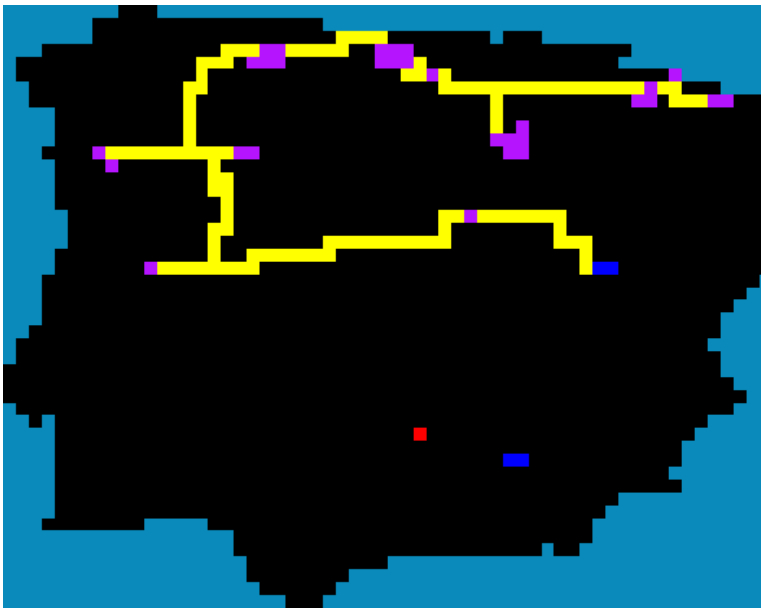
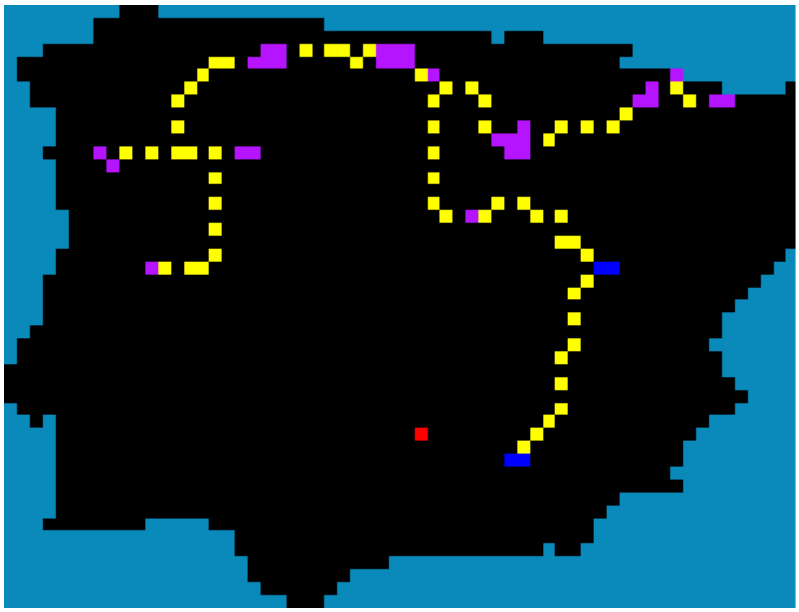


Figura 6.4: 3 espécies. "Stepping stones"



Capítulo 7

Conclusões

A dissertação introduziu uma variante do problema da floresta de Steiner com pesos nos nós, consistindo na restrição dos nós de V utilizáveis para ligar terminais do tipo k a subconjuntos $V^k \subset V$.

Foram apresentadas três formulações de programação mista inteira, duas baseadas em fluxos, que permitem obter soluções para o problema, e uma que recorre a cortes, tendo um número exponencial de restrições. No caso das formulações com fluxos provou-se que a segunda formulação é mais forte que a primeira.

Um dos objectivos do trabalho desenvolvido foi obter heurísticas que permitissem obter soluções para problemas de grande dimensão. A motivação para este objectivo resultou do trabalho, desenvolvido com outros colegas, para desenhar corredores ecológicos, com características climáticas semelhantes, entre as áreas protegidas da Península Ibérica, classificadas de acordo com critérios climáticos.

Foram desenvolvidas três heurísticas: uma designada por tipo a tipo que obtém soluções para cada um dos tipos e faz a união destas para obter a solução final; outra que é uma generalização de uma heurística primal-dual para o problema da floresta de Steiner e finalmente uma heurística do tipo GRASP.

Para o caso do problema da floresta de Steiner com custos nos nós, provou-se que a heurística PD de Demaine et al. [2009] tem um factor de aproximação ao óptimo de 2 no caso de grafos em grelha cruzada. Demaine et al. [2009] tinham provado um factor de aproximação ao óptimo de 6 para o caso de grafos planares.

As experiências computacionais, com dados simulados e com os dados da Península Ibérica, mostraram que a heurística do tipo GRASP obtém melhores resultados em todos os tipos de grafos testados. Mesmo para o caso da floresta de Steiner com pesos nos nós, a heurística GRASP obtém resultados comparáveis aos da heurística de KR [Klein and Ravi, 1995] no caso de grafos em grelha cruzada.

Como resultado da dissertação foi desenvolvida uma aplicação para obter soluções para o problema estudado, estando publicamente disponível para todos os investigadores interessados. Foi publicado um artigo com a descrição da aplicação e das heurísticas que utiliza [Brás et al., 2013], que se traduziu em algum interesse na visita à página "WEB" que contém a aplicação e no seu descarregamento.

Bibliografia

A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

D. Alagador and J.O. Cerdeira. Designing spatially-explicit ecological reserve networks in the presence of mandatory sites. *Biological Conservation*, 137: 254 – 262, 2007.

D. Alagador, M. Triviño, J. O. Cerdeira, R. Brás, M. Cabeza, and M. B. Araújo. Linking like with like: optimizing connectivity between environmentally-similar habitats. *Landscape Ecology*, 27(2):291–301, 2012.

Y. P. Aneja. An integer linear programming approach to the Steiner problem in graphs. *Networks*, 10(2):167–178, 1980.

S. Angelopoulos. The node-weighted Steiner problem in graphs of restricted node weights. In Lars Arge and Rusins Freivalds, editors, *Algorithm Theory – SWAT 2006*, volume 4059 of *Lecture Notes in Computer Science*, pages 208–219. Springer Berlin / Heidelberg, 2006.

P. Berman and V. Ramaiyer. Improved approximations for the Steiner tree problem. *Journal of Algorithms*, 17:381–408, 1994.

C. Bettstetter, J. Klinglmayr, and S. Lettner. On the degree distribution of k-connected random networks. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–6, 2010.

N. Betzler. Steiner tree problems in the analysis of biological networks. Diploma thesis, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 2006.

T.M. Brooks, R.A. Mittermeier, C.G. Mittermeier, A.B. Rylands G.A.B. da Fonseca and, W.R. Konstant, P. Flick, J. Pilgrim, S. Oldfield, G. Ma-

gin, and C. Hilton-Taylor. Habitat loss and extinction in the hotspots of biodiversity. *Conservation Biology*, 16:909–923, 2002.

R. Brás, J. O. Cerdeira, D. Alagador, and M. B. Araújo. Linking habitats for multiple species. *Environmental Modelling & Software*, 40:336 – 339, 2013.

J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. An improved lp-based approximation for Steiner tree. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 583–592. ACM, 2010.

C. Carroll, B.H. McRae, and A. Brookes. Use of linkage mapping and centrality analysis across habitat gradients to conserve connectivity of gray wolf populations in western north america. *Conservation Biology*, 26:78–87, 2012.

J. O. Cerdeira, K. J. Gaston, and L. S. Pinto. Connectivity in priority area selection for conservation. *Environmental Modeling and Assessment*, 10: 193–192, 2005.

J.O. Cerdeira, L.S. Pinto, M. Cabeza, and K.J. Gaston. Species specific connectivity in reserve-network design using graphs. *Biological Conservation*, 143:408 – 415, 2010.

R. Courant and H. Robbins. *What is mathematics?* Oxford University Press, 1979. An elementary approach to ideas and methods.

G. B. Dantzig, L. R. Ford, and D. R. Fulkerson. A primal-dual algorithm for linearprograms. In H.W. Kuhn and editors A.W. Tucker, editors, *Linear Inequalities and RelatedSystems*, pages 171–181. Princeton University Press, Princeton, NJ, 1956, 1956.

E. Demaine, M. Hajiaghayi, and P. Klein. Node-weighted Steiner tree and group Steiner tree in planar graphs. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming*, volume 5555 of *Lecture Notes in Computer Science*, pages 328–340. Springer Berlin / Heidelberg, 2009.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269 – 271, 1959.

S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

L. M. A. Drummond, M. Santos, and E. Uchoa. A distributed dual ascent algorithm for Steiner problems in multicast routing. *Networks*, 53(2):170–183, 2009.

C. W. Duin and A. Volgenant. Some generalizations of the Steiner problem in graphs. *Networks*, 17(3):353–364, 1987.

P. Erdős and A. Rényi. On random graphs I. *Publicationes Mathematicae*, 6:290–297, 1959.

P. Erdős and A. Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 7: 17–61, 1960.

U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

T. A. Feo and M. G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2): 67–71, 1989.

T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109 – 133, 1995.

B. Fuchs, W. Kern, and X. Wang. Speeding up the Dreyfus–Wagner algorithm for minimum Steiner trees. *Mathematical Methods of Operations Research*, 66:117–125, 2007.

T. Fuller and S. Sarkar. Lqgraph: a software package for optimizing connectivity in conservation planning. *Environmental Modelling and Software*, 21:750–755, 2006.

T. Fuller, M. Mungua, M. Mayfield, V. Sánchez-Cordero, and S. Sarkar. Incorporating connectivity into conservation planning: a multi-criteria case study from central mexico. *Biological Conservation*, 133:131 –142, 2006.

E. N. Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30 (4):1141–1144, 1959.

M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.

M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In D. S. Hochbaum, editor, *Approximation algorithms for NP-Hard problems*, pages 144–191. PWS Publishing, Boston, MA, 1997.

S. Guha and S. Khuller. Improved methods for approximating node weighted Steiner trees and connected dominating sets. *Information and Computation*, 150:57–74, 1999.

S. Guha, A. Moss, J. Naor, and B. Schieber. Efficient recovery from power outage. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 574–582, 1999.

S. L. Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1 (2):113–133, 1971.

I. Hanski. *The shrinking world: Ecological consequences of habitat loss*. Excellence in Ecology, 14. International Ecology Institute, Oldendorf/Luhe, Germany, 2005.

S. Hougardy and H. J. Prömel. A 1.598 approximation algorithm for the Steiner problem in graphs. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '99, pages 448–453, Philadelphia, PA, USA, 1999.

R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

M. Karpinsky and A. Zelikovsky. New approximation algorithms for the Steiner tree problems. *Journal of Combinatorial Optimization*, 1:47–65, 1997.

P. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. *Journal of Algorithms*, 19(1):104–115, 1995.

L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica*, 15:141–145, 1981.

K. J. Lai, C. P. Gomes, M. K. Schwartz, K. S. McKelvey, D. E. Calkin, and C. A. Montgomery. The Steiner multigraph problem: Wildlife corridor design for multiple species. In *AAAI'11*, pages 1357–1364, 2011.

E.L. Landguth, B.K. Hand, J. Glassy, S.A. Cushman, and M.A. Sawaya. Unicor: a species connectivity and corridor network simulator. *Ecography*, 35:9–14, 2012.

E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart,and Winston, 1976.

T. L. Magnanti and S. Raghavan. Strong formulations for network design problems with connectivity requirements. *Networks*, 45(2):61–79, 2005.

B.H. McRae and P. Beier. Circuit theory predicts gene flow in plant and animal populations. *Proceedings of the National Academy of Sciences*, 104:19885–19890, 2007.

B.H. McRae and V.B. Shah. Circuitscape user guide. *The University of California, Santa Barbara, CA, USA*, 2011.

G. Merriam. Connectivity: a fundamental ecological characteristic of landscape pattern. In J. Brandt and P. Agger, editors, *Proceedings of the 1st international seminar on methodology in landscape ecological research and planning*, pages 5–15. Roskilde Univ, Denmark, 1984.

H. J. Prömel and A. Steger. RNC-approximation algorithms for the Steiner problem. In Rüdiger Reischuk and Michel Morvan, editors, *STACS 97*, volume 1200 of *Lecture Notes in Computer Science*, pages 559–570. Springer Berlin / Heidelberg, 1997.

V. J. Rayward-Smith. The computation of nearly minimal Steiner trees in graphs. *International Journal of Mathematical Education in Science and Technology*, 14(1):15–23, 1983.

V. J. Rayward-Smith and A. Clare. On finding Steiner vertices. *Networks*, 16(3):283–294, 1986.

G. Robins and A. Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM journal on discrete mathematics*, 19(1):122–134, 2005. A preliminary version appeared in SODA 2000 pg. 770-779.

S. Saura and J. Torné. Conefor sensinode 2.2: a software package for quantifying the importance of habitat patches for landscape connectivity. *Environmental Modelling and Software*, 24:135–139, 2009.

A. Segev. The node-weighted Steiner tree problem. *Networks*, 17(1):1–17, 1987.

J. Sessions. Solving for habitat connections as a Steiner network problem. *Forest Science*, 38(1):203–207, 1992.

H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Japonica*, 24(6):573–577, 1980.

V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2003.

P. Winter. Steiner problem in networks: A survey. *Networks*, 17(2):129–167, 1987.

R. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28:271–287, 1984.

X. Xu, Y. Wang, H. Du, P. J. Wan, F. Zou, X. Li, and W. Wu. Approximations for node-weighted Steiner tree in unit disk graphs. *Optimization Letters*, 4:405–416, 2010.

A. Zelikovsky. An 11/6-approximation algorithm for the network Steiner problem. *Algorithmica*, 9:463–470, 1993.

A. Zelikovsky. Better approximation bounds for the network and euclidian Steiner tree problems. Technical report, CS-96-06, University of Virginia, Charlottesville, VA, 1996.

F. Zou, X. Li, D. Kim, and W. Wu. Two constant approximation algorithms for node-weighted Steiner tree in unit disk graphs. In Boting Yang, Ding-Zhu Du, and Cao Wang, editors, *Combinatorial Optimization and Applications*, volume 5165 of *Lecture Notes in Computer Science*, pages 278–285. Springer Berlin / Heidelberg, 2008.

H. Önal and R.A. Briers. Incorporating spatial criteria in optimum reserve network selection. *Proceedings of the Royal Society of London Series B-Biological Sciences*, 269, 2002.

H. Önal and R.A. Briers. Selection of minimum-boundary reserve network. *Proceedings of the Royal Society of London Series B-Biological Sciences*, 270:1487–1491, 2003.

H. Önal and R.A. Briers. Designing a conservation reserve network with minimal fragmentation: a linear integer programming approach. *Environmental Modelling and Assessment*, 10:193 – 202, 2005.